



US012001801B2

(12) **United States Patent**  
**Wang et al.**

(10) **Patent No.: US 12,001,801 B2**  
(45) **Date of Patent: Jun. 4, 2024**

(54) **QUESTION ANSWERING USING DYNAMIC  
QUESTION-ANSWER DATABASE**

(71) Applicant: **salesforce.com, inc.**, San Francisco, CA  
(US)

(72) Inventors: **Yuanxin Wang**, Thornhill (CA);  
**Anuprit Kale**, El Cerrito, CA (US);  
**Zachary Alexander**, Berkeley, CA  
(US); **Na Cheng**, Dublin, CA (US)

(73) Assignee: **Salesforce, Inc.**, San Francisco, CA  
(US)

(\*) Notice: Subject to any disclaimer, the term of this  
patent is extended or adjusted under 35  
U.S.C. 154(b) by 590 days.

(21) Appl. No.: **16/685,909**

(22) Filed: **Nov. 15, 2019**

(65) **Prior Publication Data**  
US 2021/0149964 A1 May 20, 2021

(51) **Int. Cl.**  
**G06F 40/30** (2020.01)  
**G06F 16/9032** (2019.01)  
**G06F 18/22** (2023.01)  
**G06F 40/205** (2020.01)  
**G06N 20/00** (2019.01)

(52) **U.S. Cl.**  
CPC ..... **G06F 40/30** (2020.01); **G06F 16/9032**  
(2019.01); **G06F 18/22** (2023.01); **G06F**  
**40/205** (2020.01); **G06N 20/00** (2019.01)

(58) **Field of Classification Search**  
None  
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,577,188 A	11/1996	Zhu
5,608,872 A	3/1997	Schwartz et al.
5,649,104 A	7/1997	Carleton et al.
5,715,450 A	2/1998	Ambrose et al.
5,761,419 A	6/1998	Schwartz et al.
5,819,038 A	10/1998	Carleton et al.
5,821,937 A	10/1998	Tonelli et al.
5,831,610 A	11/1998	Tonelli et al.
5,873,096 A	2/1999	Lim et al.
5,918,159 A	6/1999	Fomukong et al.

(Continued)

OTHER PUBLICATIONS

“Google Plus Users”, Google+Ripples, Oct. 31, 2011 [retrieved on  
Feb. 21, 2012 from Internet at <http://www.googleplusers.com/google-ripples.html>], 3 pages

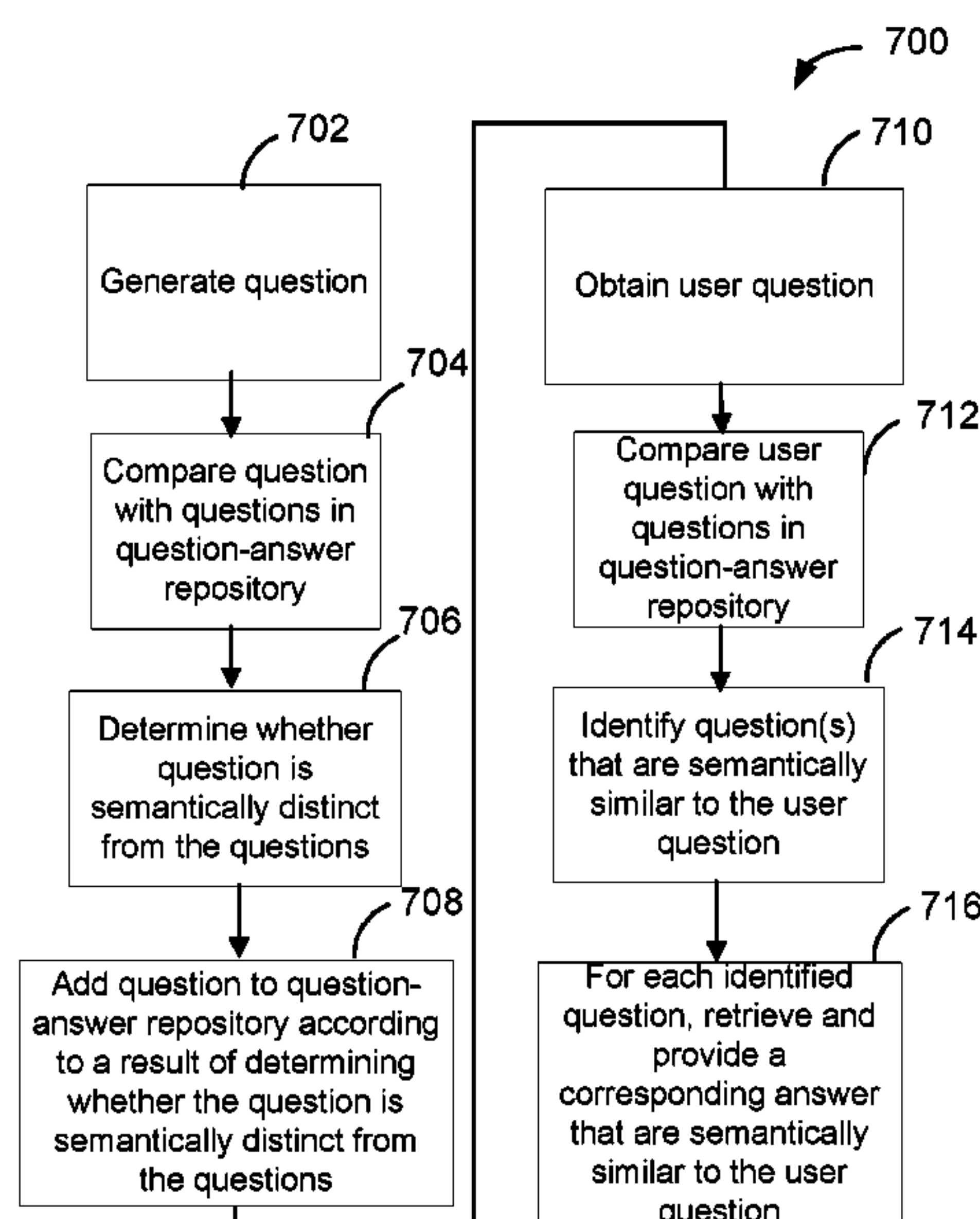
*Primary Examiner* — Thu N Nguyen

(74) *Attorney, Agent, or Firm* — Weaver Austin  
Villeneuve & Sampson LLP

(57) **ABSTRACT**

Disclosed are some implementations of systems, apparatus,  
methods and computer program products for integrating  
question generation and answer retrieval in a question  
answer system. The system generates a question using a set  
of documents and determines whether it is semantically  
distinct from questions in a question-answer repository.  
After determining that the question is semantically distinct  
from questions in the question-answer repository, the system  
adds the question to the question-answer repository. Upon  
receipt of a user-submitted question, the system uses the  
question-answer repository to identify a semantically similar  
question. The system retrieves an answer corresponding to  
the identified question from the question-answer repository  
and provides the answer in response to the user-submitted  
question.

**20 Claims, 10 Drawing Sheets**





(56)

## References Cited

## U.S. PATENT DOCUMENTS

5,963,953 A	10/1999	Cram et al.	7,508,789 B2	3/2009	Chan
5,983,227 A	11/1999	Nazem et al.	7,599,935 B2	10/2009	La Rotonda et al.
6,092,083 A	7/2000	Brodersen et al.	7,603,331 B2	10/2009	Tuzhilin et al.
6,161,149 A	12/2000	Achacoso et al.	7,603,483 B2	10/2009	Psounis et al.
6,169,534 B1	1/2001	Raffel et al.	7,620,655 B2	11/2009	Larsson et al.
6,178,425 B1	1/2001	Brodersen et al.	7,644,122 B2	1/2010	Weyer et al.
6,189,011 B1	2/2001	Lim et al.	7,668,861 B2	2/2010	Steven
6,216,133 B1	4/2001	Masthoff	7,698,160 B2	4/2010	Beaven et al.
6,216,135 B1	4/2001	Brodersen et al.	7,730,478 B2	6/2010	Weissman
6,233,617 B1	5/2001	Rothwein et al.	7,747,648 B1	6/2010	Kraft et al.
6,236,978 B1	5/2001	Tuzhilin	7,779,039 B2	8/2010	Weissman et al.
6,266,669 B1	7/2001	Brodersen et al.	7,779,475 B2	8/2010	Jakobson et al.
6,288,717 B1	9/2001	Dunkle	7,827,208 B2	11/2010	Bosworth et al.
6,295,530 B1	9/2001	Ritchie et al.	7,853,881 B1	12/2010	Aly Assal et al.
6,324,568 B1	11/2001	Diec et al.	7,945,653 B2	5/2011	Zukerberg et al.
6,324,693 B1	11/2001	Brodersen et al.	8,005,896 B2	8/2011	Cheah
6,336,137 B1	1/2002	Lee et al.	8,014,943 B2	9/2011	Jakobson
D454,139 S	3/2002	Feldcamp et al.	8,015,495 B2	9/2011	Achacoso et al.
6,367,077 B1	4/2002	Brodersen et al.	8,032,297 B2	10/2011	Jakobson
6,393,605 B1	5/2002	Loomans	8,073,850 B1	12/2011	Hubbard et al.
6,405,220 B1	6/2002	Brodersen et al.	8,082,301 B2	12/2011	Ahlgren et al.
6,411,949 B1	6/2002	Schaffer	8,095,413 B1	1/2012	Beaven
6,434,550 B1	8/2002	Warner et al.	8,095,531 B2	1/2012	Weissman et al.
6,446,089 B1	9/2002	Brodersen et al.	8,095,594 B2	1/2012	Beaven et al.
6,535,909 B1	3/2003	Rust	8,103,611 B2	1/2012	Tuzhilin et al.
6,549,908 B1	4/2003	Loomans	8,150,913 B2	4/2012	Cheah
6,553,563 B2	4/2003	Ambrose et al.	8,209,308 B2	6/2012	Rueben et al.
6,560,461 B1	5/2003	Fomukong et al.	8,209,333 B2	6/2012	Hubbard et al.
6,574,635 B2	6/2003	Stauber et al.	8,275,836 B2	9/2012	Beaven et al.
6,577,726 B1	6/2003	Huang et al.	8,457,545 B2	6/2013	Chan
6,601,087 B1	7/2003	Zhu et al.	8,484,111 B2	7/2013	Frankland et al.
6,604,117 B2	8/2003	Lim et al.	8,490,025 B2	7/2013	Jakobson et al.
6,604,128 B2	8/2003	Diec et al.	8,504,945 B2	8/2013	Jakobson et al.
6,609,150 B2	8/2003	Lee et al.	8,510,045 B2	8/2013	Rueben et al.
6,621,834 B1	9/2003	Scherpbier et al.	8,510,664 B2	8/2013	Rueben et al.
6,654,032 B1	11/2003	Zhu et al.	8,566,301 B2	10/2013	Rueben et al.
6,665,648 B2	12/2003	Brodersen et al.	8,646,103 B2	2/2014	Jakobson et al.
6,665,655 B1	12/2003	Warner et al.	10,528,916 B1 *	1/2020	Taylor ..... G06Q 50/20
6,684,438 B2	2/2004	Brodersen et al.	10,956,822 B1 *	3/2021	Kern ..... G06N 20/00
6,711,565 B1	3/2004	Subramaniam et al.	2001/0044791 A1	11/2001	Richter et al.
6,724,399 B1	4/2004	Katchour et al.	2002/0072951 A1	6/2002	Lee et al.
6,728,702 B1	4/2004	Subramaniam et al.	2002/0082892 A1	6/2002	Raffel et al.
6,728,960 B1	4/2004	Loomans et al.	2002/0129352 A1	9/2002	Brodersen et al.
6,732,095 B1	5/2004	Warshavsky et al.	2002/0140731 A1	10/2002	Subramaniam et al.
6,732,100 B1	5/2004	Brodersen et al.	2002/0143997 A1	10/2002	Huang et al.
6,732,111 B2	5/2004	Brodersen et al.	2002/0162090 A1	10/2002	Parnell et al.
6,754,681 B2	6/2004	Brodersen et al.	2002/0165742 A1	11/2002	Robbins
6,763,351 B1	7/2004	Subramaniam et al.	2003/0004971 A1	1/2003	Gong
6,763,501 B1	7/2004	Zhu et al.	2003/0018705 A1	1/2003	Chen et al.
6,768,904 B2	7/2004	Kim	2003/0018830 A1	1/2003	Chen et al.
6,772,229 B1	8/2004	Achacoso et al.	2003/0066031 A1	4/2003	Laane et al.
6,782,383 B2	8/2004	Subramaniam et al.	2003/0066032 A1	4/2003	Ramachandran et al.
6,804,330 B1	10/2004	Jones et al.	2003/0069936 A1	4/2003	Warner et al.
6,826,565 B2	11/2004	Ritchie et al.	2003/0070000 A1	4/2003	Coker et al.
6,826,582 B1	11/2004	Chatterjee et al.	2003/0070004 A1	4/2003	Mukundan et al.
6,826,745 B2	11/2004	Coker	2003/0070005 A1	4/2003	Mukundan et al.
6,829,655 B1	12/2004	Huang et al.	2003/0074418 A1	4/2003	Coker et al.
6,842,748 B1	1/2005	Warner et al.	2003/0120675 A1	6/2003	Stauber et al.
6,850,895 B2	2/2005	Brodersen et al.	2003/0151633 A1	8/2003	George et al.
6,850,949 B2	2/2005	Warner et al.	2003/0159136 A1	8/2003	Huang et al.
6,907,566 B1	6/2005	McElfresh et al.	2003/0187921 A1	10/2003	Diec et al.
7,062,502 B1	6/2006	Kesler	2003/0189600 A1	10/2003	Gune et al.
7,069,231 B1	6/2006	Cinarkaya et al.	2003/0204427 A1	10/2003	Gune et al.
7,069,497 B1	6/2006	Desai	2003/0206192 A1	11/2003	Chen et al.
7,100,111 B2	8/2006	McElfresh et al.	2003/0225730 A1	12/2003	Warner et al.
7,181,758 B1	2/2007	Chan	2004/0001092 A1	1/2004	Rothwein et al.
7,269,590 B2	9/2007	Hull et al.	2004/0010489 A1	1/2004	Rio et al.
7,289,976 B2	10/2007	Kihneman et al.	2004/0015981 A1	1/2004	Coker et al.
7,340,411 B2	3/2008	Cook	2004/0027388 A1	2/2004	Berg et al.
7,356,482 B2	4/2008	Frankland et al.	2004/0128001 A1	7/2004	Levin et al.
7,373,599 B2	5/2008	McElfresh et al.	2004/0186860 A1	9/2004	Lee et al.
7,401,094 B1	7/2008	Kesler	2004/0193510 A1	9/2004	Catahan et al.
7,406,501 B2	7/2008	Szeto et al.	2004/0199489 A1	10/2004	Barnes-Leon et al.
7,412,455 B2	8/2008	Dillon	2004/0199536 A1	10/2004	Barnes-Leon et al.
7,454,509 B2	11/2008	Boulter et al.	2004/0199543 A1	10/2004	Braud et al.
			2004/0249854 A1	12/2004	Barnes-Leon et al.
			2004/0260534 A1	12/2004	Pak et al.
			2004/0260659 A1	12/2004	Chan et al.
			2004/0268299 A1	12/2004	Lei et al.

(56)

References Cited

U.S. PATENT DOCUMENTS

2005/0050555 A1

3/2005

Exley et al.

2005/0091098 A1

4/2005

Brodersen et al.

2008/0249972 A1

10/2008

Dillon

2009/0063415 A1

3/2009

Chatfield et al.

2009/0100342 A1

4/2009

Jakobson

2009/0177744 A1

7/2009

Marlow et al.

2011/0218958 A1

9/2011

Warshavsky et al.

2011/0247051 A1

10/2011

Bulumulla et al.

2012/0042218 A1

2/2012

Cinarkaya et al.

2012/0233137 A1

9/2012

Jakobson et al.

2012/0290407 A1

11/2012

Hubbard et al.

2013/0212497 A1

8/2013

Zelenko et al.

2013/0218948 A1

8/2013

Jakobson

2013/0218949 A1

8/2013

Jakobson

2013/0218966 A1

8/2013

Jakobson

2013/0247216 A1

9/2013

Cinarkaya et al.

2013/0304730 A1 \*

11/2013

Zhou ..... G06Q 30/02707/723

2014/0297571 A1 \*

10/2014

Beamon ..... G06F 16/313706/46

2014/0359537 A1

12/2014

Jakobson et al.

2015/0006289 A1

1/2015

Jakobson et al.

2015/0007050 A1

1/2015

Jakobson et al.

2015/0095162 A1

4/2015

Jakobson et al.

2015/0142596 A1

5/2015

Jakobson et al.

2015/0172563 A1

6/2015

Jakobson et al.

2017/0351401 A1

12/2017

Pascale et al.

2017/0351781 A1

12/2017

Alexander et al.

2018/0089585 A1

3/2018

Rickard, Jr. et al.

2018/0096250 A1

4/2018

Rickard, Jr. et al.

2018/0096372 A1

4/2018

Rickard, Jr. et al.

2018/0101534 A1

4/2018

Alexander, Jr. et al.

2018/0101537 A1

4/2018

Govindarajan et al.

2018/0239924 A1

8/2018

Rickard, Jr. et al.

2018/0293241 A1

10/2018

Chittar et al.

2019/0005089 A1

1/2019

Kempf et al.

2020/0065511 A1

2/2020

Rickard, Jr. et al.

2020/0097496 A1

3/2020

Alexander et al.

2020/0097544 A1

3/2020

Alexander et al.

2020/0097563 A1

3/2020

Alexander et al.

2020/0097608 A1

3/2020

Xiu et al.

2020/0097809 A1

3/2020

Velasco et al.

2020/0117671 A1

4/2020

Govindarajan et al.

2020/0233874 A1

7/2020

Chittar et al.

2020/0242305 A1 \*

7/2020

Wu ..... G06N 5/02

2020/0334546 A1 \*

10/2020

Muschett ..... G06N 5/025

2021/0140021 A1

5/2021

Frost et al.

2021/0149921 A1

5/2021

Alexander et al.

2021/0149933 A1

5/2021

Chang et al.

2021/0149949 A1

5/2021

Alexander et al.

2021/0149964 A1

5/2021

Wang et al.

2021/0150144 A1

5/2021

Kale et al.

2021/0150146 A1

5/2021

Alexander et al.

2021/0150610 A1

5/2021

Mao et al.

2021/0248261 A1

8/2021

Rickard et al.

\* cited by examiner



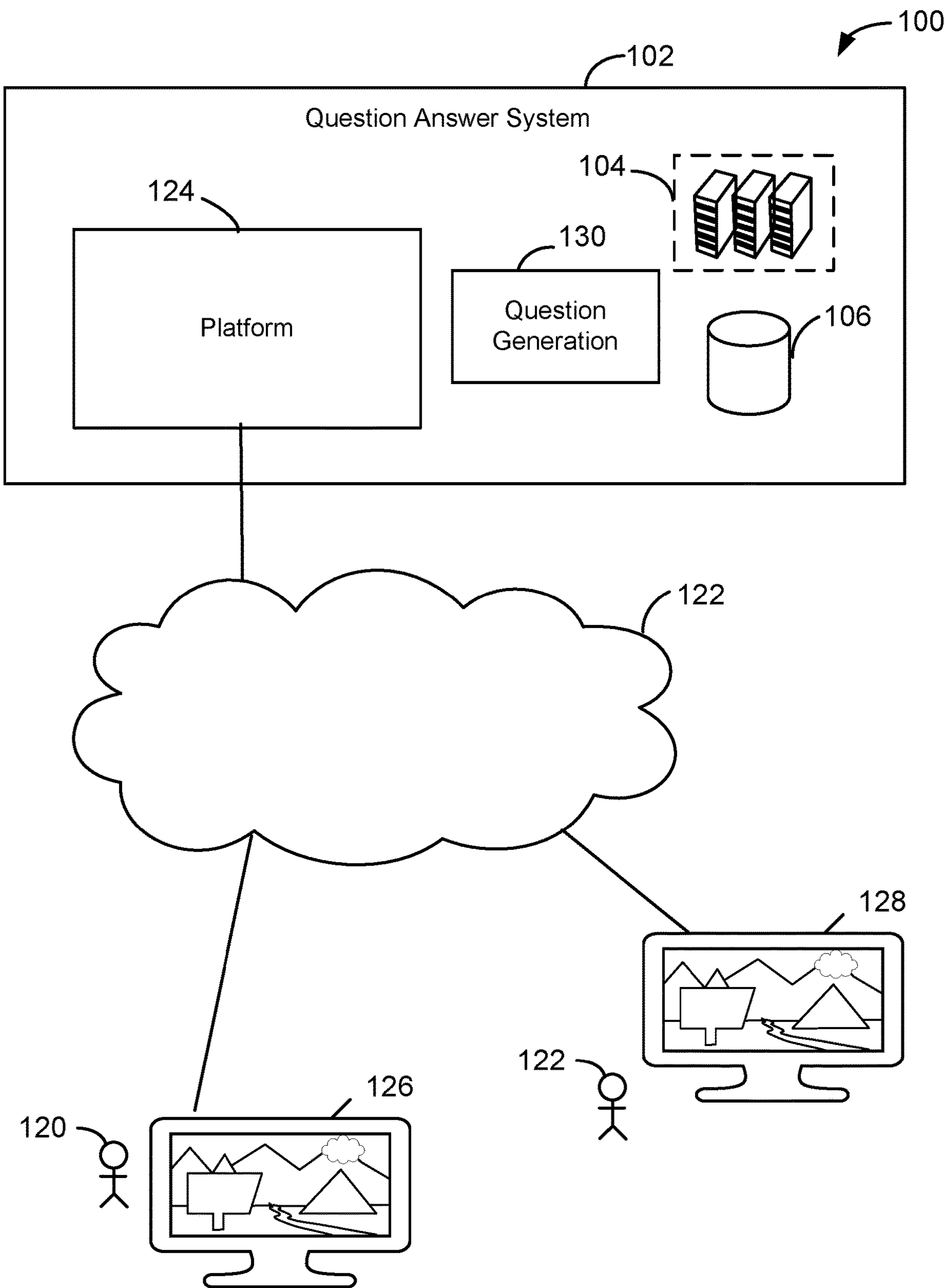


Figure 1

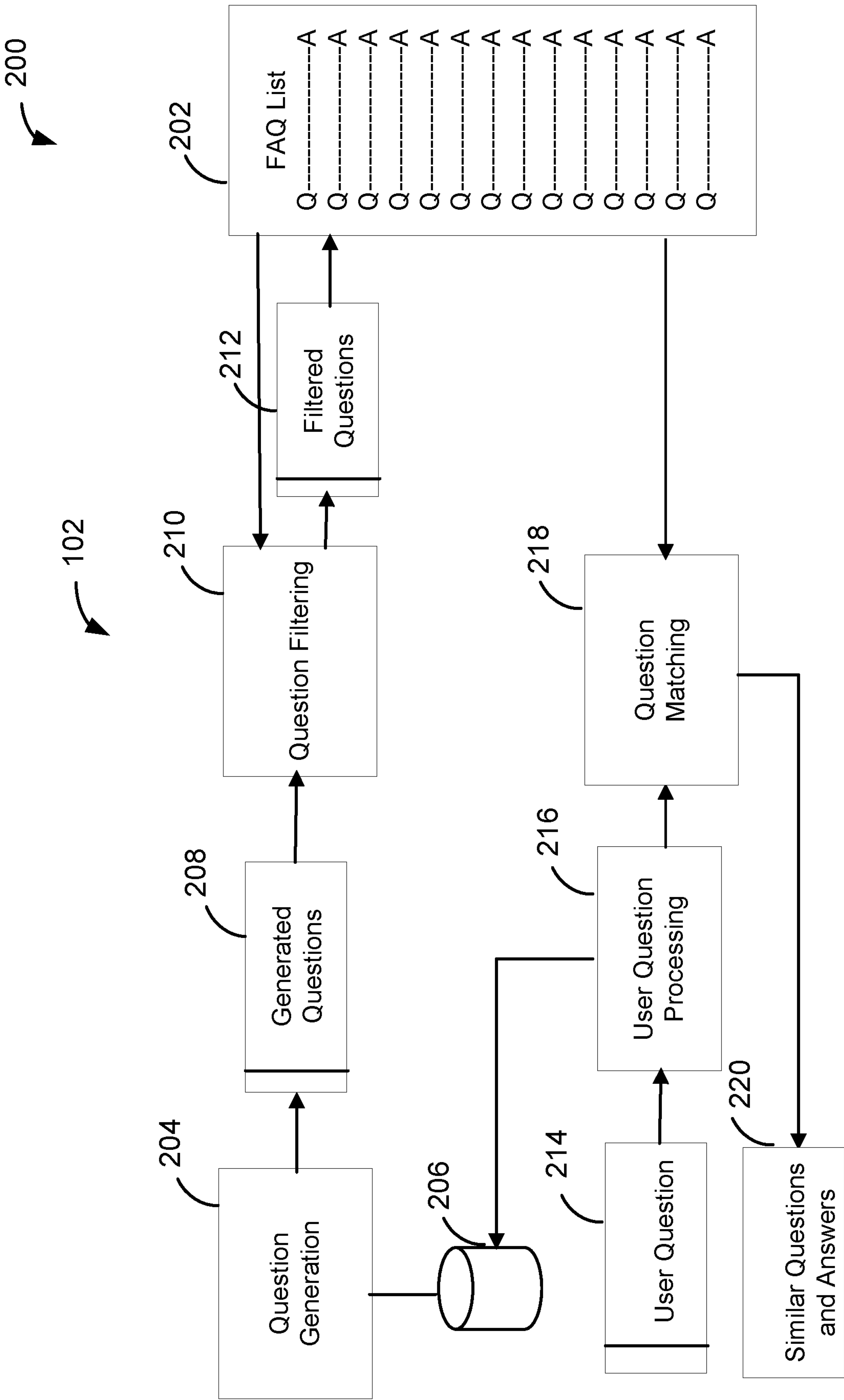
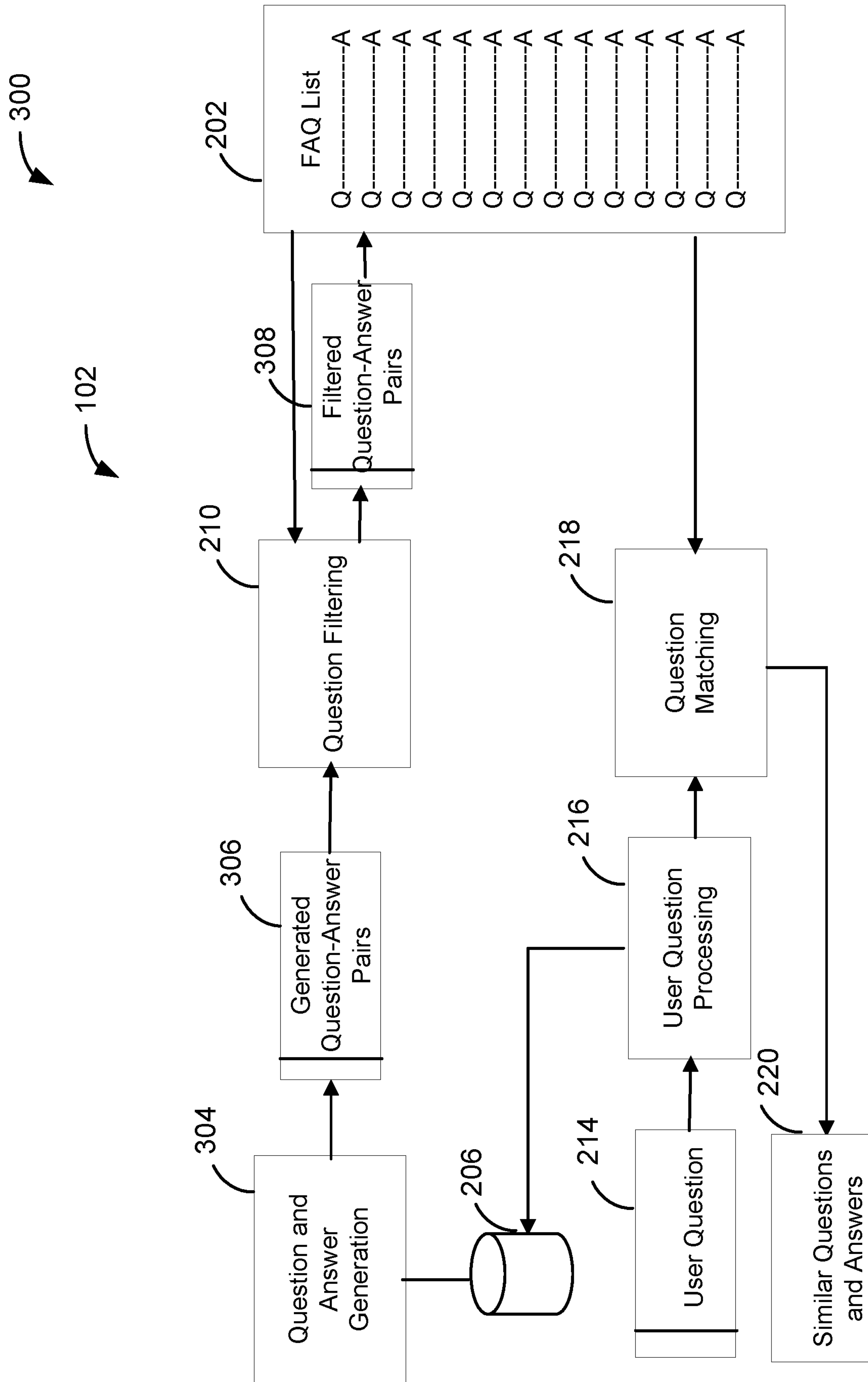


Figure 2



### Figure 3

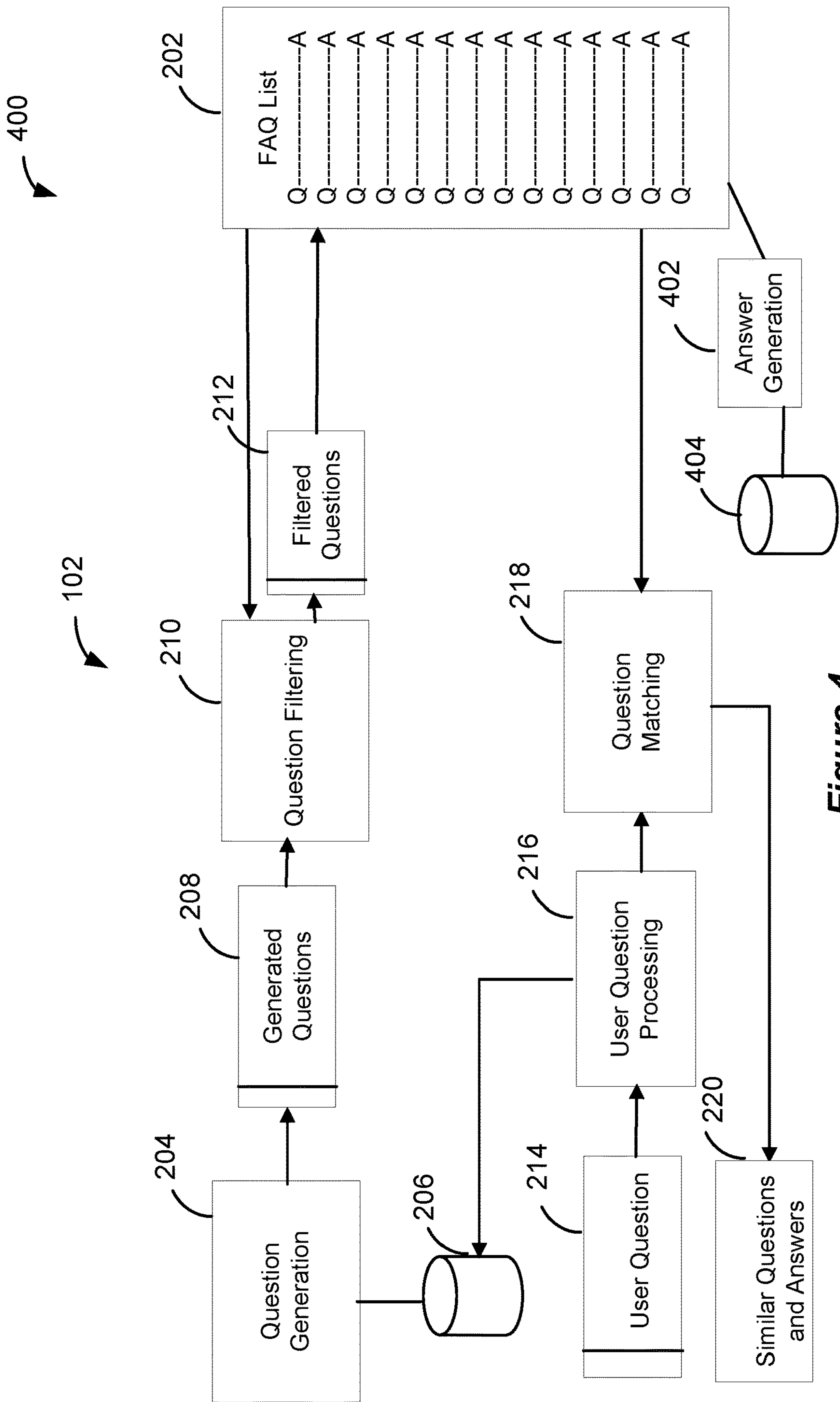


Figure 4

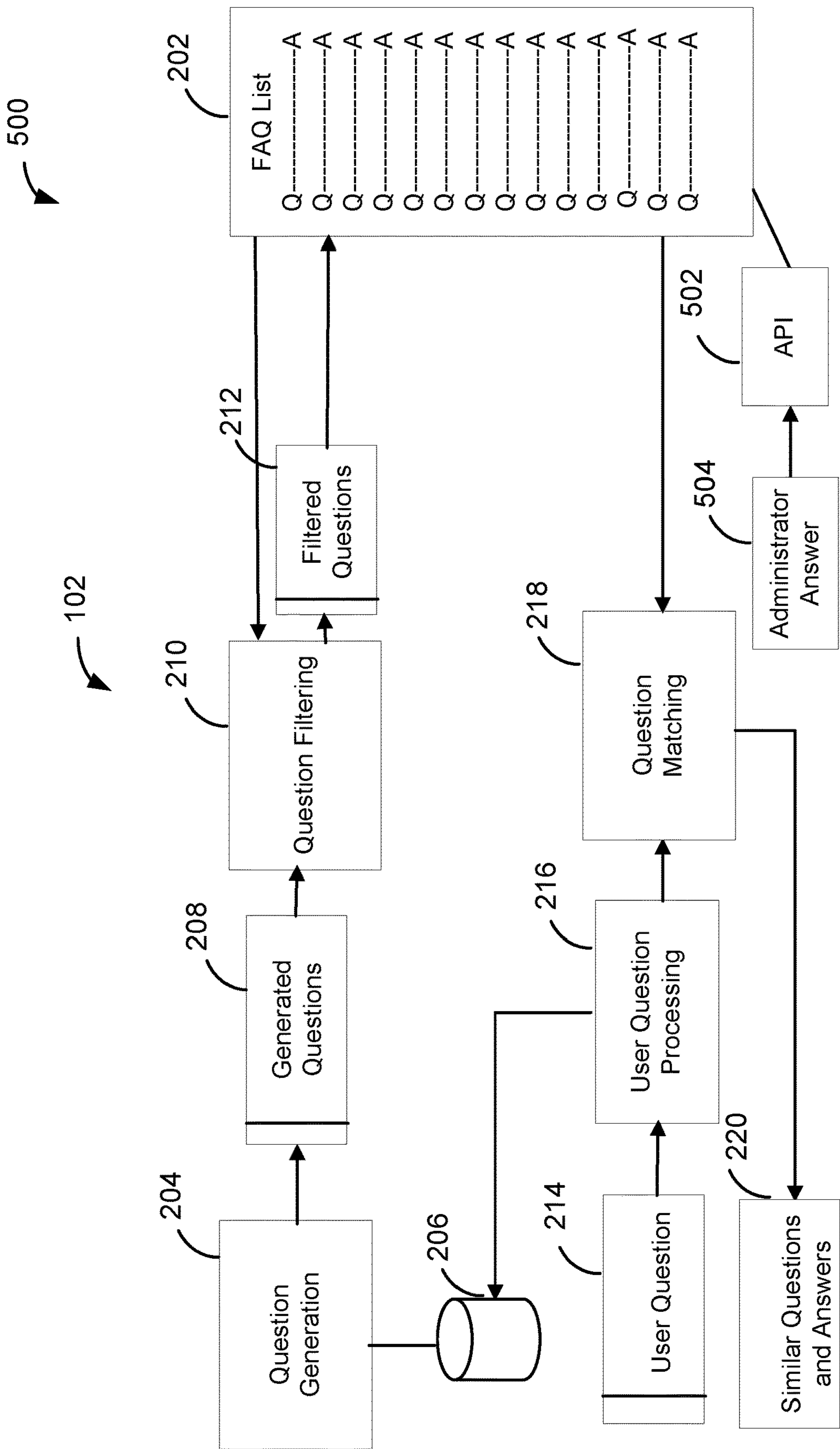


Figure 5



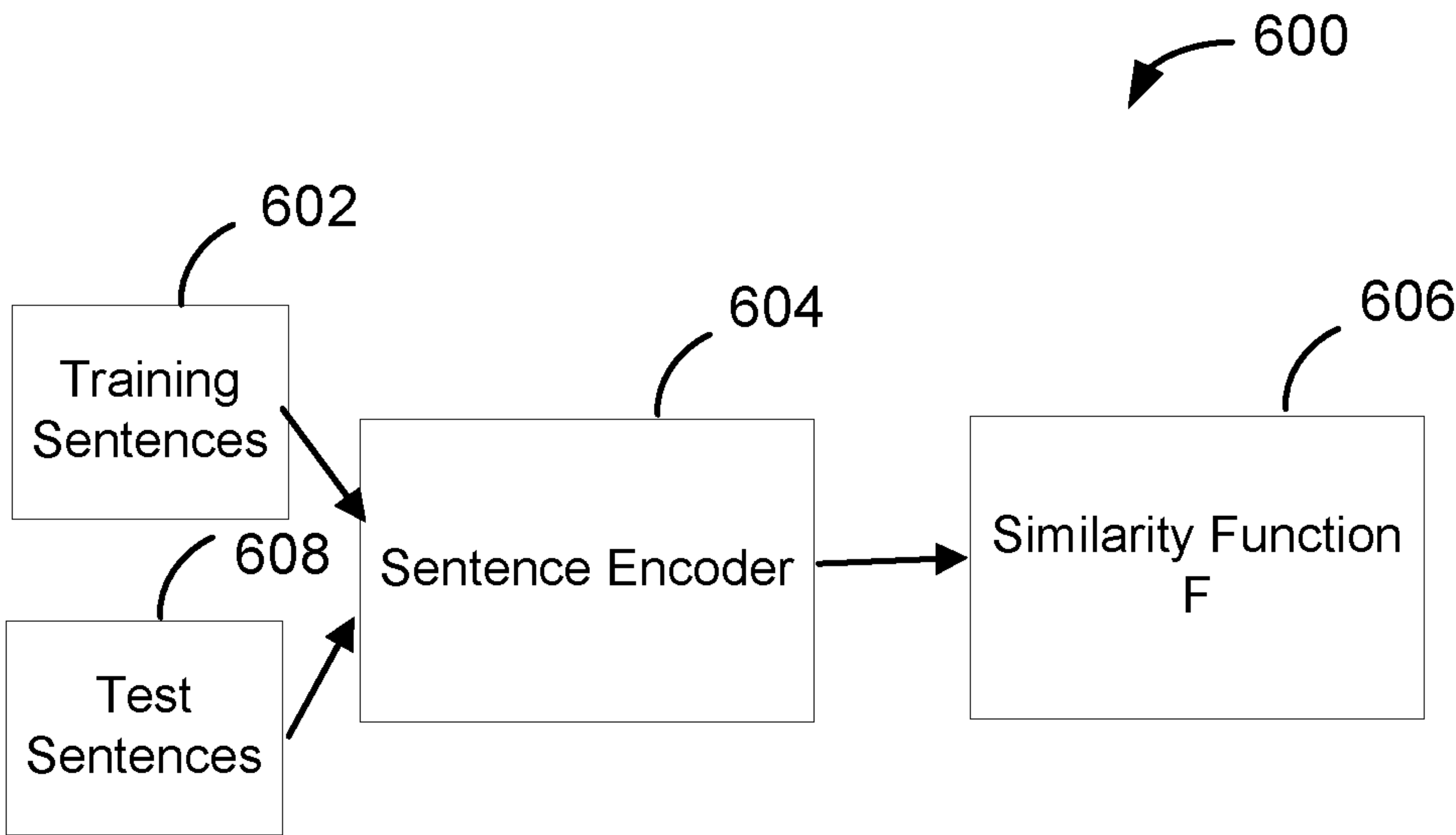


Figure 6A

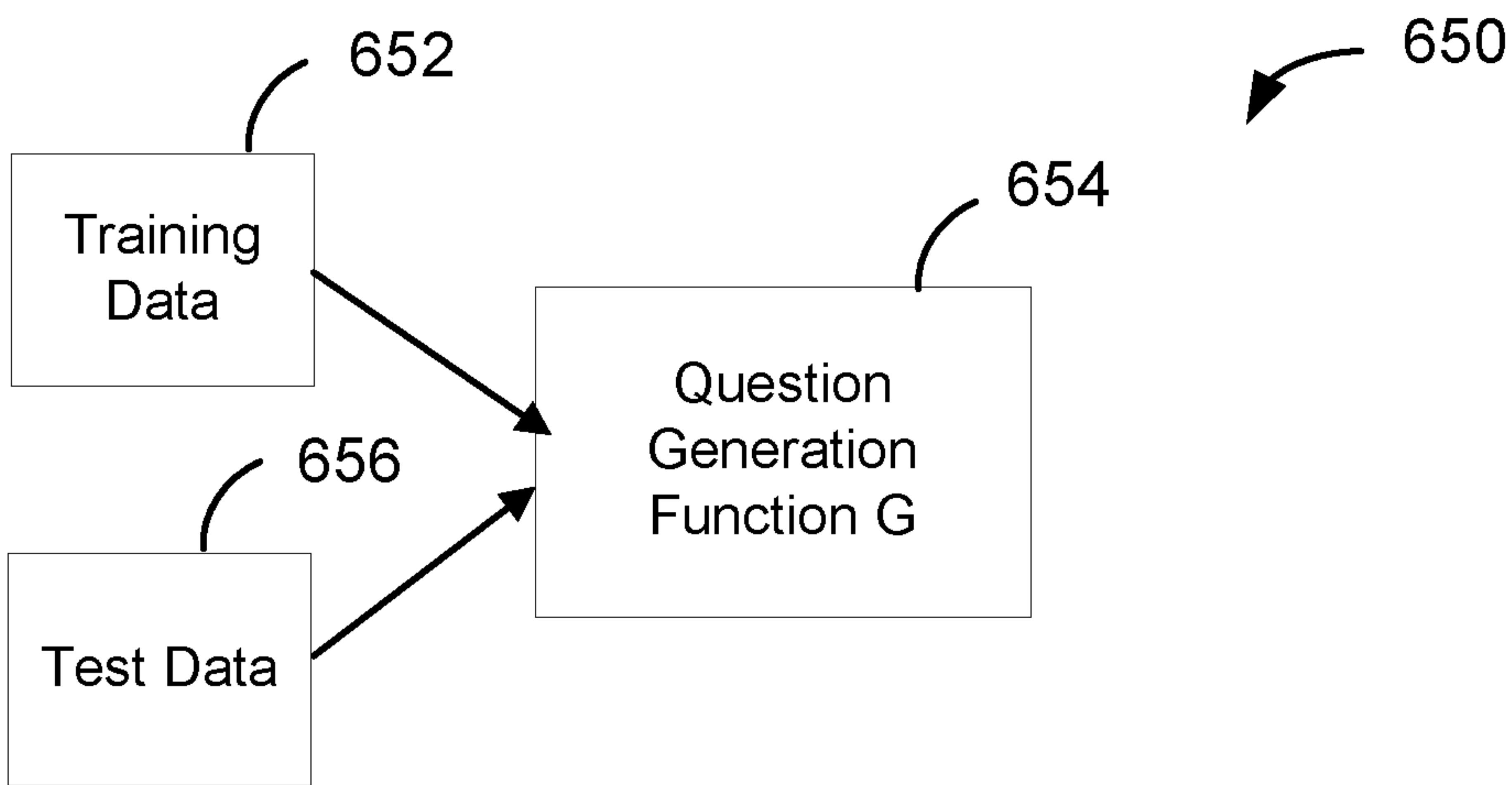
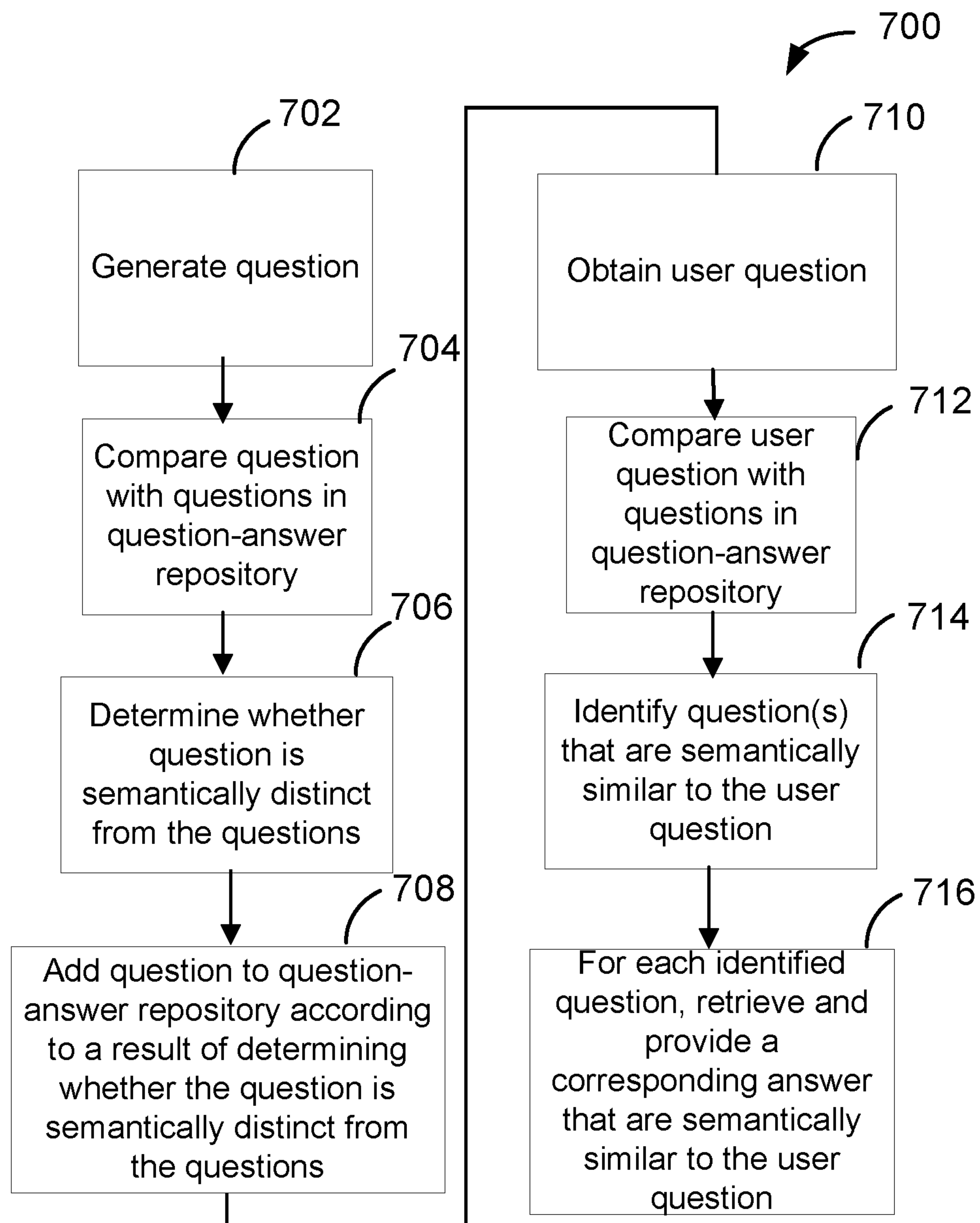


Figure 6B

**Figure 7**

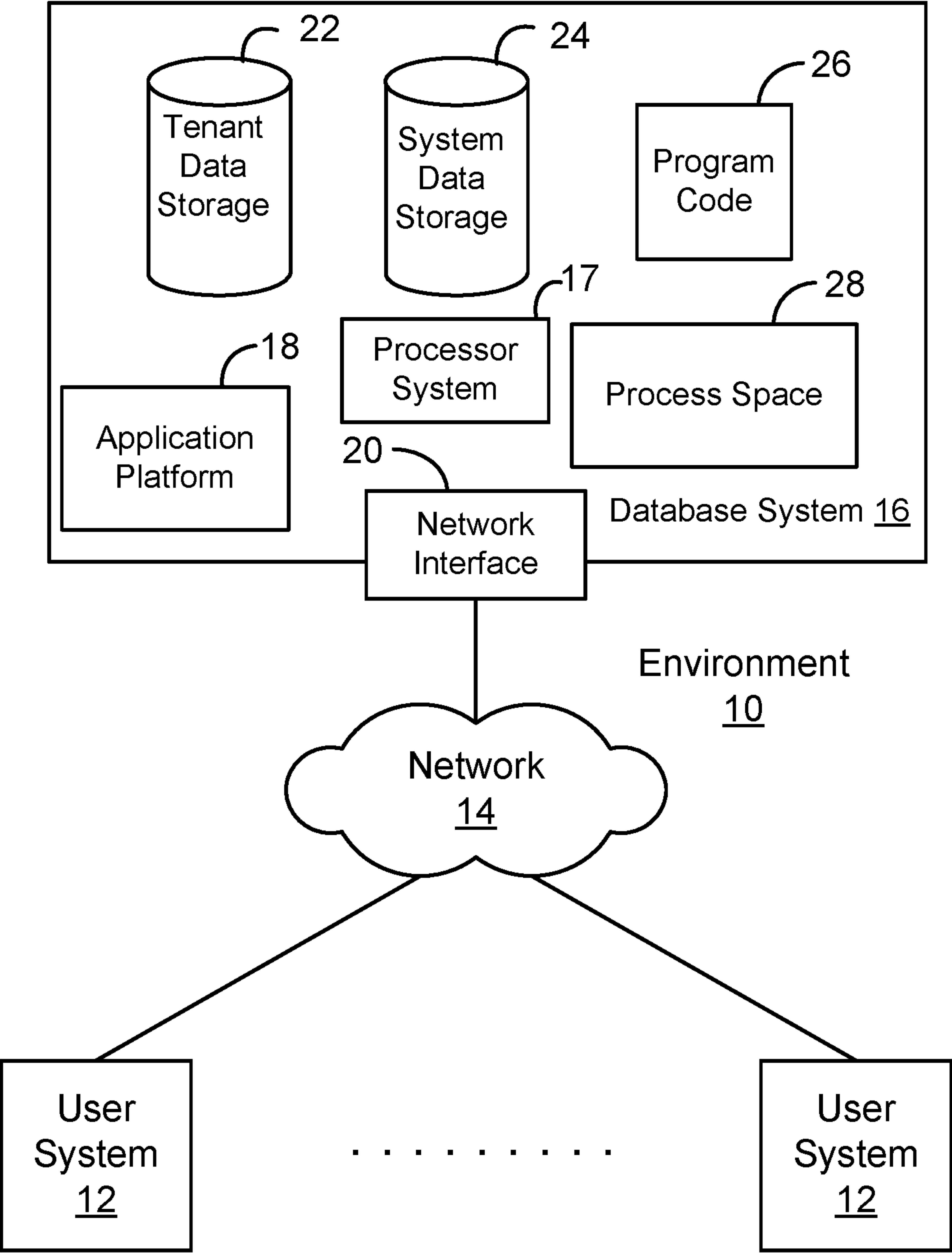


Figure 8A



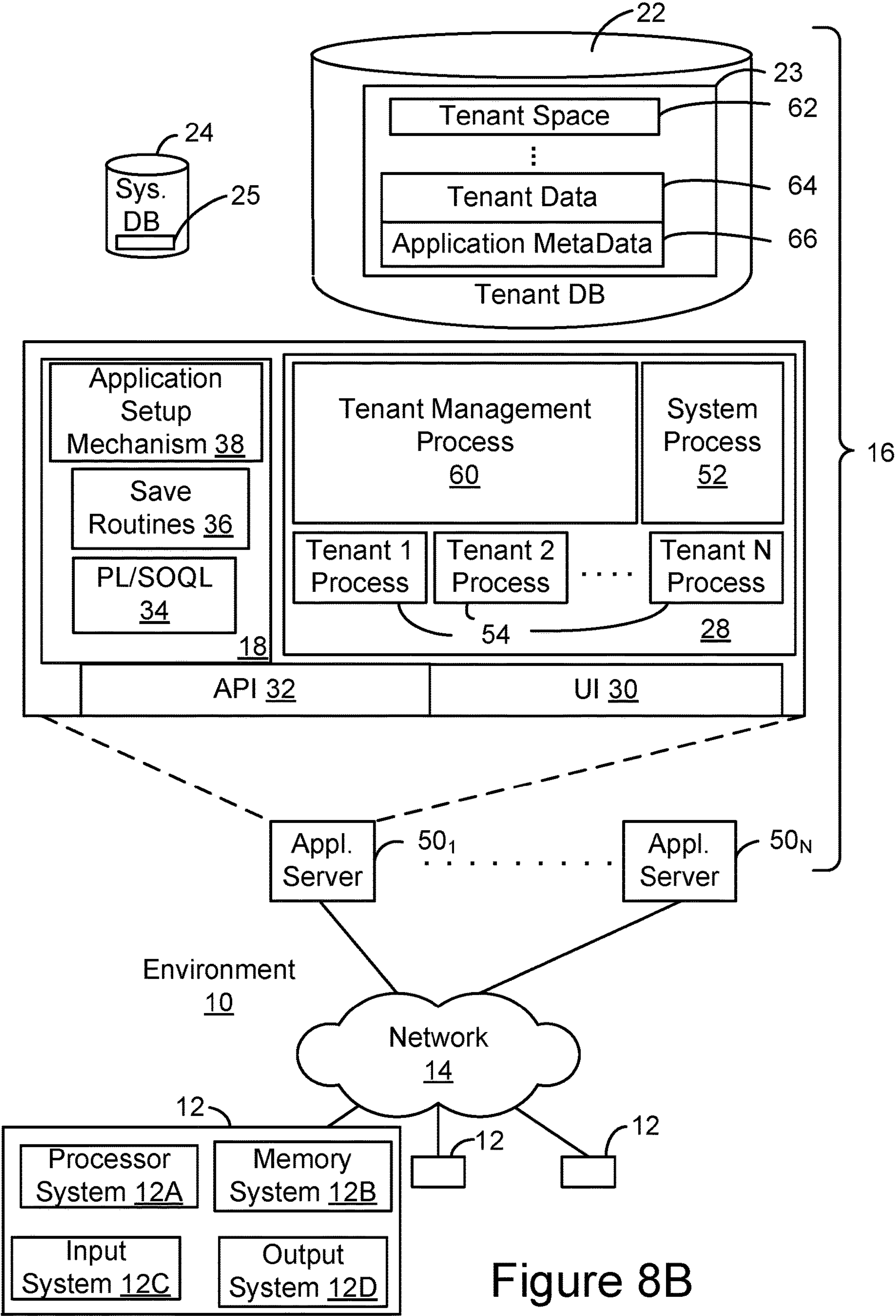


Figure 8B

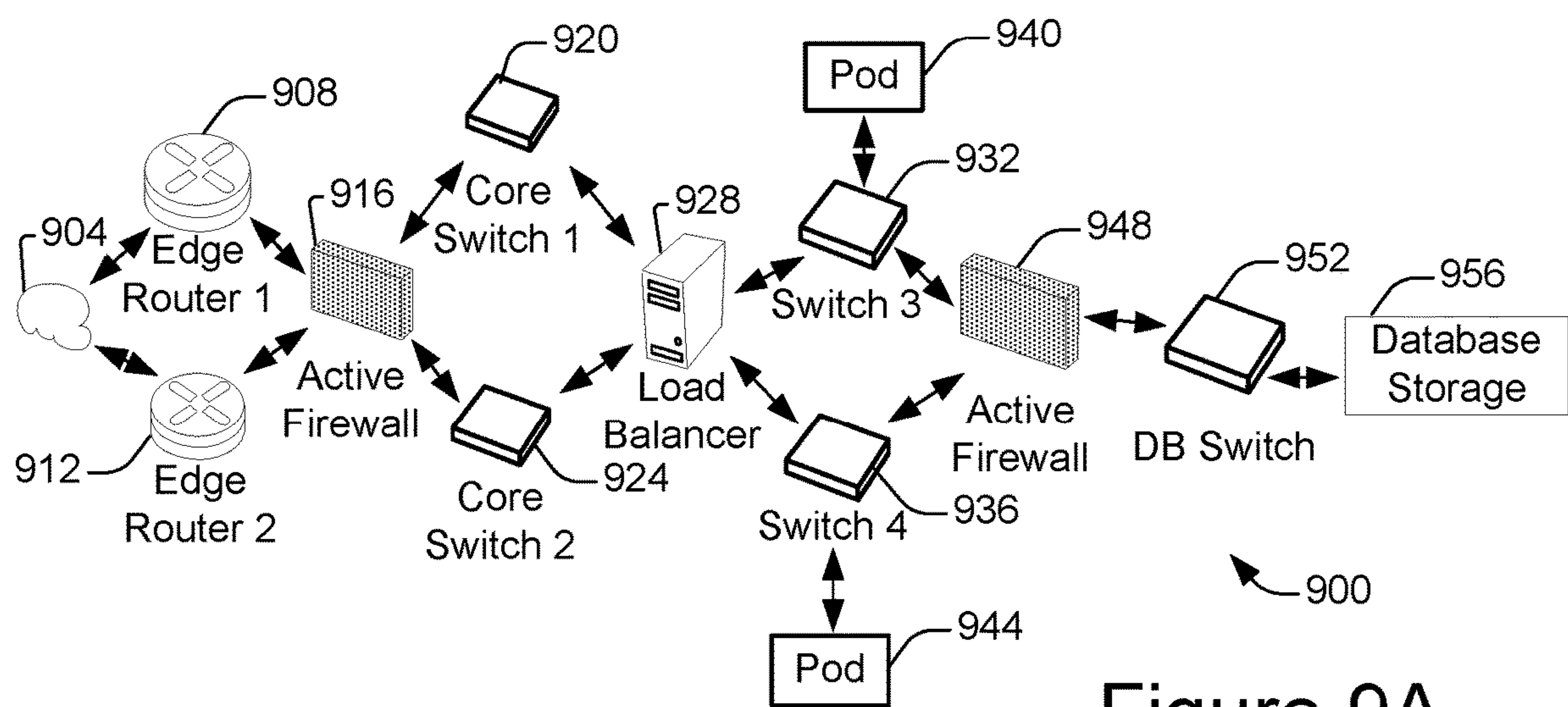


Figure 9A

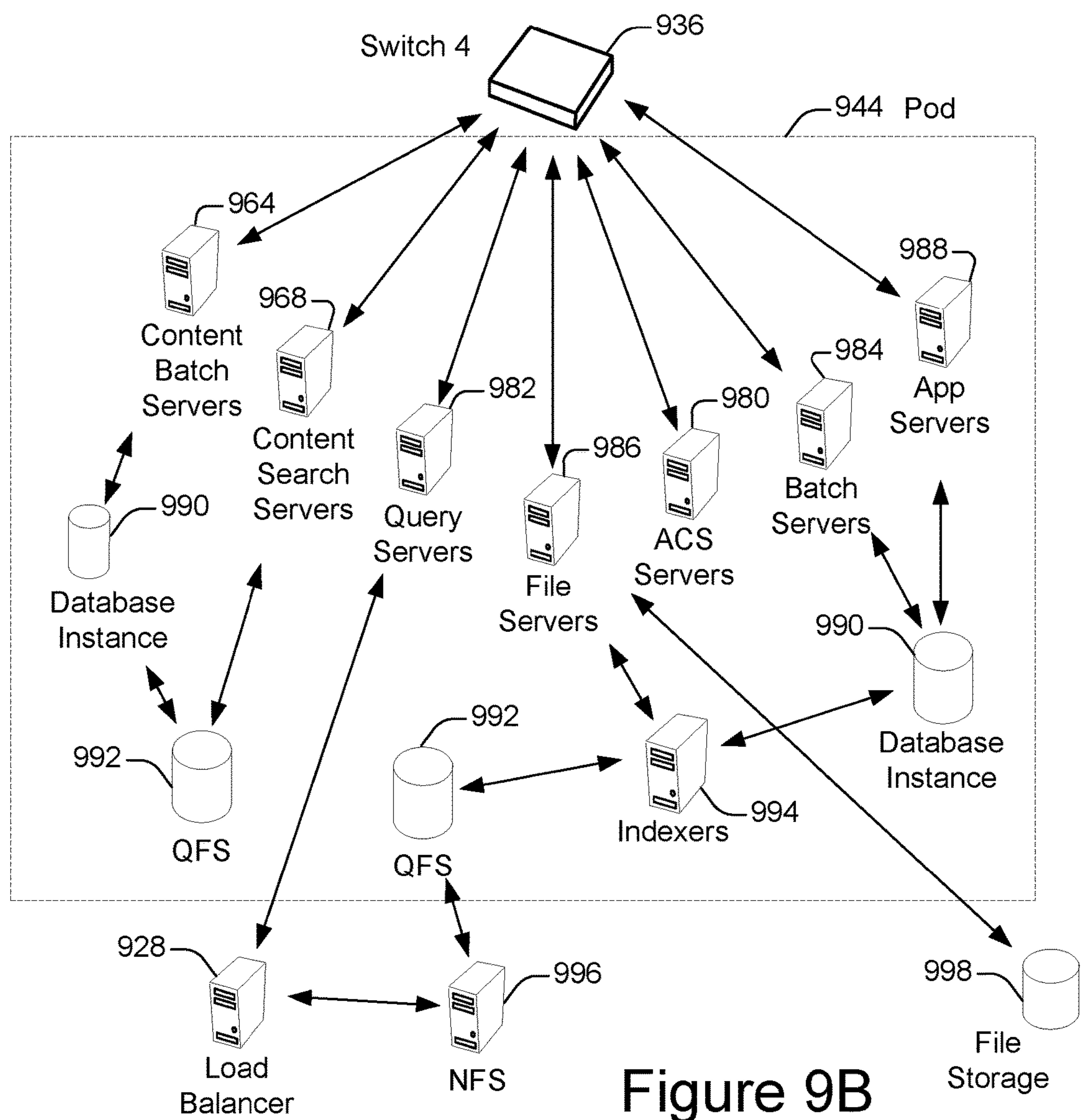


Figure 9B



## QUESTION ANSWERING USING DYNAMIC QUESTION-ANSWER DATABASE

### COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material, which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the United States Patent and Trademark Office patent file or records but otherwise reserves all copyright rights whatsoever.

### TECHNICAL FIELD

This patent document generally relates to systems and techniques associated with automatic question answering technology. More specifically, this patent document discloses techniques for integrating question generation and answer retrieval using a dynamic question-answer repository.

### BACKGROUND

“Cloud computing” services provide shared network-based resources, applications, and information to computers and other devices upon request. In cloud computing environments, services can be provided by servers to users’ computer systems via the Internet and wireless networks rather than installing software locally on users’ computer systems. A user can interact with social networking systems, question-answer systems, electronic mail (email) systems, and instant messaging systems, by way of example, in a cloud computing environment.

Question answer systems automatically answer questions posed by humans in a natural language. These systems typically implement question answering technology that queries a structured database of knowledge or information. Some question answer systems query an unstructured collection of natural language documents to provide answers to user-submitted questions.

### BRIEF DESCRIPTION OF THE DRAWINGS

The included drawings are for illustrative purposes and serve only to provide examples of possible structures and operations for the disclosed systems, apparatus, methods and computer program products for leveraging and managing assessment environments in an assessment hub. These drawings in no way limit any changes in form and detail that may be made by one skilled in the art without departing from the spirit and scope of the disclosed implementations.

FIG. 1 shows a system diagram of an example of a system 100 in which a question answer system is implemented, in accordance with some implementations.

FIG. 2 shows a system diagram of a system 200 including an example question answer system in which questions are system-generated, in accordance with some implementations.

FIG. 3 shows a system diagram of a system 300 including another example question answer system in which questions and corresponding answers are system-generated, in accordance with some implementations.

FIG. 4 shows a system diagram of a system 400 including another example question answer system in which answers are system-generated, in accordance with some implementations.

FIG. 5 shows a system diagram of a system 500 including another example question answer system in which answers are submitted for unanswered questions, in accordance with some implementations.

FIG. 6A shows a diagram 600 illustrating the training of a similarity function, in accordance with some implementations.

FIG. 6B shows a diagram 650 illustrating the training of a question generation function, in accordance with some implementations.

FIG. 7 shows an example of a method 700 for implementing a question answer system, in accordance with some implementations.

FIG. 8A shows a block diagram of an example of an environment 10 in which an on-demand database service can be used in accordance with some implementations.

FIG. 8B shows a block diagram of an example of some implementations of elements of FIG. 8A and various possible interconnections between these elements.

FIG. 9A shows a system diagram of an example of architectural components of an on-demand database service environment 900, in accordance with some implementations.

FIG. 9B shows a system diagram further illustrating an example of architectural components of an on-demand database service environment, in accordance with some implementations.

### DETAILED DESCRIPTION

Examples of systems, apparatus, methods and computer program products according to the disclosed implementations are described in this section. These examples are being provided solely to add context and aid in the understanding of the disclosed implementations. It will thus be apparent to one skilled in the art that implementations may be practiced without some or all of these specific details. In other instances, certain operations have not been described in detail to avoid unnecessarily obscuring implementations. Other applications are possible, such that the following examples should not be taken as definitive or limiting either in scope or setting.

In the following detailed description, references are made to the accompanying drawings, which form a part of the description and in which are shown, by way of illustration, specific implementations. Although these implementations are described in sufficient detail to enable one skilled in the art to practice the disclosed implementations, it is understood that these examples are not limiting, such that other implementations may be used and changes may be made without departing from their spirit and scope. For example, the operations of methods shown and described herein are not necessarily performed in the order indicated. It should also be understood that the methods may include more or fewer operations than are indicated. In some implementations, operations described herein as separate operations may be combined. Conversely, what may be described herein as a single operation may be implemented in multiple operations.

Some implementations of the disclosed systems, apparatus, methods and computer program products are configured for answering user-submitted questions using a dynamically updated question-answer repository. In some implementations, the question-answer repository is updated in parallel with answer retrieval. The question-answer repository may be updated to include additional questions such that redundant questions are minimized or eliminated.



In some implementations, a question generation process is trained and implemented to generate questions using a set of documents. Question generation can include parsing the set of documents to extract question(s) from the set of documents. In addition, question generation can include parsing the set of documents to generate question(s) using text extracted from the set of documents.

In some implementations, for each question that is generated, the question is compared with questions in a question-answer repository. Questions that are determined to be semantically distinct from the questions in the question-answer repository are added to the question-answer repository. Any questions that are determined not to be semantically distinct from the questions in the question-answer repository are not added to the question-answer repository.

In some implementations, a similarity function is trained to compare semantic similarity between two different segments of text. For each text segment pair, the similarity function generates a score that indicates an amount of similarity between the two text segments. The similarity function can be applied during question generation and/or answer retrieval.

In some implementations, system-generated questions are compared with questions in a question-answer repository using a similarity function. More particularly, a system-generated question is compared with questions in the question-answer repository such that a plurality of similarity scores are generated, where each of the similarity scores represents a similarity between the system-generated question and a corresponding one of the stored questions. The system may determine whether the system-generated question is semantically distinct from the questions based, at least in part, on the similarity scores. The system-generated question is then added to the question-answer repository according to a result of determining whether the question is semantically distinct from the questions in the question-answer repository.

In some implementations, the question-answer repository is updated to include an answer corresponding to the system-generated question. The question-answer repository may be updated automatically using an answer extracted or generated by the question answer system. Alternatively, the question-answer repository may be updated using an answer submitted via a user such as a system administrator.

In some implementations, answer retrieval is facilitated using a similarity function. Specifically, a user-submitted question is compared with questions within a question-answer repository such that a plurality of similarity scores are generated, where each of the similarity scores represents a similarity between the user-submitted question and a corresponding one of the stored questions. One or more questions that are semantically similar to the user-submitted question may be identified using the similarity scores. For each semantically similar question, a corresponding answer may be provided in response to the user-submitted question. In some implementations, a question-answer pair that includes the semantically similar question and the corresponding answer is provided for each semantically similar question.

In some implementations, question answer system updates the set of documents used to generate questions based, at least in part, on user-submitted questions. This can be performed in real-time upon submission of user questions or in a "batch" process. In some instances, documents such as chat transcripts and/or log data can be updated to include user-submitted questions. The system may periodically process these documents or associated updates to generate

questions. Therefore, the set of documents that the system processes to generate questions and/or answers can include documents such as chat transcript, log data, blogs, articles, web pages, and/or social media posts.

In some implementations, the question answer system makes a determination as to whether to update the question-answer repository to include the user-submitted question using a similarity function. Specifically, a similarity function may be applied to determine whether the user-submitted question is semantically distinct from questions in the question-answer repository. In the event that the user-submitted question is determined to be semantically distinct from questions in the question-answer repository, the question-answer repository may be updated to include the user-submitted question. The question answer system may exclude user-submitted questions that are determined not to be semantically distinct from being stored in the question-answer repository.

Question answer systems typically query a structured database that stores information that can be queried. Sometimes, the database includes question-answer pairs that each includes a question and corresponding answer. The database is typically static and generated by a system administrator. Since the database is static, the amount of information stored in the database is limited. Moreover, the generation of the database relies upon the knowledge or resources of the administrator. As a result, information maintained in the database is typically incomplete. Therefore, answers that are provided to users are often inadequate, resulting in user disappointment and frustration.

In accordance with various implementations, a question answer system updates a question-answer repository that is used for answer retrieval. By dynamically expanding the question-answer repository, answers that are retrieved are more likely to be accurate and responsive to user-submitted questions. Moreover, by incorporating system-generated questions and user-submitted questions, expansion of the question-answer repository may be integrated with answer retrieval. The question-answer repository may be updated by a process that may be performed online or offline. Therefore, the question-answer repository may be updated in parallel with answer retrieval.

By way of illustration, Claire is an employee at an organization, Acme Construction, Inc. Claire accesses a question answer system via her computing device. The question answer system periodically parses documents to expand a question-answer repository by storing additional system-generated semantically distinct questions in the question-answer repository. One of the questions the system generates includes the question, "Instructions for reporting a customer issue?" An administrator periodically reviews the question-answer repository and adds answers for questions that do not have a corresponding answer. In this case, the administrator adds an answer, "Report customer issue by creating a new case using 'New Case' tab from main menu."

Clair submits a question, "How do I generate a new case?" The question-answer system queries the question-answer repository, identifies the system-generated question, "Instructions for reporting a customer issue?" and ascertains that it is semantically similar to Claire's question. The system provides the identified system-generated question and corresponding answer, "Instructions for reporting a customer issue" and "Report customer issue by creating a new case using New Case tab from main menu for display by the client device." The user's question "How do I generate a new case?" is added to a question log. The



## 5

question answer system periodically processes new documents including the question log, determines that the user's question is not semantically distinct from other questions in the question-answer repository. However, the question answer system identifies another user's question in the question log that is semantically distinct from questions in the question-answer repository and adds that user's question to the repository. An answer to the user's question may be added to the repository at the time that the user's question is added or at a later time.

FIG. 1 shows a system diagram of an example of a system 100 in which a question answer system is implemented, in accordance with some implementations. Question answer system 102 includes a variety of different hardware and/or software components that are in communication with each other. In the non-limiting example of FIG. 1, system 102 includes any number of computing devices such as servers 104. Servers 104 are in communication with one or more storage mediums 106 configured to store and maintain relevant data and/or metadata used to perform some of the techniques disclosed herein, as well as to store and maintain relevant data and/or metadata generated by the techniques disclosed herein. Storage mediums 106 may further store computer-readable instructions configured to perform some of the techniques described herein. In some implementations, storage mediums 106 can store Frequently Asked Questions (FAQ) list(s), which may be generated, updated, and accessed as described in further detail below.

In some implementations, system 102 is configured to store user profiles/user accounts associated with users of system 102. Information maintained in a user account of a user can include or indicate a user email address and credentials of the user. For example, credentials of the user can include a username and password. The information can also include the name of the user. In addition, the information can indicate a role of the user and associated permissions. For example, an employee may have access to answer retrieval services, while an administrator may have access to answer retrieval services and read-write privileges for the FAQ list(s).

Client devices 126, 128 may be in communication with system 102 via network 122. More particularly, client devices 126, 128 may communicate with servers 104 via network 122. For example, network 122 can be the Internet. In another example, network 122 comprises one or more local area networks (LAN) in communication with one or more wide area networks (WAN) such as the Internet.

Embodiments described herein are often implemented in a cloud computing environment, in which network 122, servers 104, and possible additional apparatus and systems such as multi-tenant databases may all be considered part of the "cloud." Servers 104 may be associated with a network domain, such as www.salesforce.com and may be controlled by a data provider associated with the network domain. A user of client computing device 126 can have an account at salesforce.com®. By logging into this account, the user can access the various question answer services provided by servers 104.

In some implementations, users 120, 122 of client devices 126, 128 can access question answer services by logging into system 102 via platform 124. More particularly, client devices 126, 128 can log into system 102 via an application programming interface (API) or via a graphical user interface (GUI) using credentials of corresponding users 120, 122, respectively.

Users 120, 122 can submit questions via question answer system 102, which retrieves pertinent answers from FAQ

## 6

list(s) and provides the answers via client devices 126, 128. Examples of devices used by users include, but are not limited to a desktop computer or portable electronic device such as a smartphone, a tablet, a laptop, a wearable device such as Google Glass®, another optical head-mounted display (OHMD) device, a smart watch, etc.

In some implementations, question answer system 102 updates FAQ list(s) stored in storage medium 106 to include additional question(s). In addition, question answer system 102 may update the FAQ list(s) to include an answer for each additional question. Updates to the FAQ list(s) may be performed automatically based, at least in part, on documents processed by question answer system 102. In addition, updates to FAQ list(s) may be performed responsive to communications received from a client device such as device 128.

System 102 may include a Question Generation Module 130 that generates questions using a question generation function, which may be trained using a set of training data. The question generation function may operate to parse documents to extract questions. In addition, the question generation function may operate to extract text segments from documents and generate questions using the text segments.

As described herein, system 102 updates FAQ list(s) to include those system-generated questions that are determined to be semantically distinct from questions in the FAQ list(s). Similarly, system 102 may filter system-generated questions that are not semantically distinct such that they are not added to the FAQ list(s). System 102 may determine whether questions are semantically distinct from each other using a similarity function.

In some implementations, system 102 generates a similarity function using training data. The similarity function may be applied during answer retrieval or updating of the FAQ list(s), as described herein. The similarity function may determine a level of similarity between two different text segments (e.g., questions). Specifically, the similarity function may generate a score that indicates semantic similarity between two different text segments.

In some implementations, system 102 provides a user such as an administrative user 122 read-write access to FAQ list(s) via client device 128 according to their corresponding role. Assuming permissions give user 122 read-write access to FAQ list(s), user 122 may access the FAQ list(s) and submits an answer for a question within the FAQ list that does not have a corresponding answer. The system may then update the corresponding database entry to include the answer submitted by user 122.

Question answer system 102 may dynamically update FAQ list(s) in parallel with answer retrieval using the FAQ list. Various examples of a question answer system 102 that can be implemented will be described in further detail below with reference to FIGS. 2-5.

FIG. 2 shows a system diagram of a system 200 including an example question answer system in which questions are system-generated, in accordance with some implementations. As shown in this example, a data object 202 such as a database may include a plurality of entries, where each of the entries specifies or otherwise indicates a question-answer pair of a FAQ list. In some implementations, the database includes a plurality of records, where each of the records corresponds to a different question. For example, the database can include a relational database having a question field and an answer field.

An administrator may cause data object 202 to be generated or initialized such that it includes a plurality of initial



entries. For example, administrator may enter or upload information including a plurality of question-answer pairs.

System **102** may perform question generation **204** using data source(s) **206** to generate questions **208**. Data source(s) **206** can include documents stored in real or virtual storage devices. In addition, data source(s) **206** can include documents accessible via a network such as the Internet. Example documents include, but are not limited to, chat transcripts, user-submitted questions, articles, blogs, social media web pages or posts, web pages, customer relationship management (CRM) records such as cases, case comments, etc.

Question generation **204** can include application of a question generation function that extracts questions from documents or otherwise generates questions using text segments that have been extracted from the documents. The question generation function can include a computer-generated model, as will be described in further detail below.

System **102** performs question filtering **210** to filter system-generated questions **208**. Specifically, for a given system-generated question **208**, system **102** may compare the system-generated question **208** with a plurality of questions stored in data object **202** (e.g., FAQ list). Question filtering **210** can be performed using a similarity function to identify filtered questions **212** that are semantically distinct from questions that are already stored in data object **202**. System **102** may update FAQ list by storing filtered questions **212** in data object **202** (e.g., by adding entries to FAQ list). For example, system **102** may update data object **202** to add a new record to a database for each semantically distinct system-generated question. In this manner, system **102** may add new entries to FAQ list to expand the number of questions in the FAQ list.

System **102** may also perform answer retrieval for a user-submitted question **214**. Specifically, system **102** performs user question processing **216**, which can include storing the user question **214** to data source(s) **206**, which enables semantically distinct user questions to be added to data object **202**. In addition, system **102** performs question matching **218** to identify relevant questions in data object **202**. More particularly, question matching **218** can include applying a similarity function that identifies one or more questions in data object **202** that are semantically similar to the user-submitted question **214**. Answers corresponding to the semantically similar questions may be retrieved from data object **202** and provided to the user in response to the user question **214**. In some implementations, semantically similar questions and their respective answers **220** may be retrieved from data object **202** and provided for presentation to the user so that the user can determine the relevance of the answer based upon the corresponding question.

In accordance with various implementations, an answer may be associated with each stored question not having a corresponding answer. Various example implementations of a question answer system will be described in further detail below with reference to FIGS. **3-5**.

In some implementations, answers may automatically generated by question answer system **102**. Automatic answer generation will be described in further detail below with reference to FIGS. **3-4**.

FIG. **3** shows a system diagram of a system **300** including another example question answer system in which questions and corresponding answers are system-generated, in accordance with some implementations. As shown in this example, question answer system **102** may perform question and answer generation **304** such that one or more question-answer pairs **306** are generated. More particularly, for each question that is system-generated, a corresponding answer

may be generated. For example, an answer to a particular question may be retrieved from documents in data source **206**. As another example, answer segments may be extracted from the documents in data source **206** and used to generate an answer for the particular question.

In some implementations, an answer generation function may be used to generate an answer. An answer generation function may perform pattern matching. For example, pattern matching may be performed in text that is in close proximity to the generated question within the documents. In some implementations, an answer generation function may be trained using training data.

In some implementations, for each system-generated question that is determined to be semantically distinct from questions in data object **202**, a corresponding question-answer pair **308** may be stored in data object **202**. For those system-generated questions that are determined to not be semantically distinct from questions in data object **202**, the corresponding question-answer pairs are not stored in data object **202**.

FIG. **4** shows a system diagram of a system **400** including another example question answer system in which answers are system-generated in accordance with some implementations. In this example, question answer system **102** performs answer generation **402** after question filtering **210** identifies filtered questions **212** that are semantically distinct from questions stored in data object **202**. More particularly, answer generation **402** may query data object **202** or another data object containing filtered questions **212** to identify each question that does not have an associated answer. The question can include a system-generated question. In some implementations, the question can include a user-submitted question.

In some implementations, answer generation **402** accesses documents in data store(s) **404**. Data store **404** may be identical to data store **206** or distinct from data store **206**. Answer generation **402** parses documents in data store **404** and generates an answer for a corresponding question not having a corresponding answer. Specifically, answer generation **402** may extract an answer from data store **404** or generate an answer from extracted text segments obtained from the documents. As discussed above, answer generation **402** can be performed for a question via an answer generation function, which may perform pattern matching based, at least in part, on the question or portion thereof. The generated answer may then be stored in association with the corresponding system-generated question in data object **202**.

In some implementations, answer generation is performed based upon input received from an administrator or other individual via a client device. FIG. **5** shows a system diagram of a system **500** including another example question answer system in which answers are submitted for unanswered questions, in accordance with some implementations. As shown in this example, question answer system **102** may provide an application programming interface (API) **502** that provides access to data object **202**. Specifically, an administrator or other individual with read-write access can login to their account via a client device and interact with a user interface object such as a tab, button, or menu. Responsive to this interaction, information including one or more questions that do not have corresponding answers may be retrieved and provided for display by the client device. For example, the system may query data object to identify questions that do not have corresponding answers and provide the questions for display by the client device. The administrator may submit an answer **504** corresponding one of the unanswered questions via a graphical user interface



(GUI) or input device. The system may then store the answer **504** in association with the corresponding unanswered question.

In the examples described above, unanswered questions are stored in the FAQ list of data object **202**. In other implementations, unanswered questions are stored in another intermediate data structure that is independent from data object **202**. For each unanswered question, an answer may be obtained and the corresponding question-answer pair may then be stored in data object **202** such that the FAQ list includes the additional question-answer pair.

FIG. **6A** shows a diagram **600** illustrating the training of a similarity function, in accordance with some implementations. As shown in this example, a set of similarity training data can include training sentences **602** that are “encoded” by a sentence encoder **604** to represent words or phrases with labels such as numerical values. Specifically, the “encoding” may be performed by assigning a label or numerical value to each word or phrase within a sentence such that the sentence is associated with a sequence of labels or numerical values. For example, labels may be used to identify semantically similar phrases such as “go shopping” and “driving to the store.” In this example, the phrases are not syntactically similar, but have similar meanings. For example, training sentences **602** can include pairs of semantically similar sentences. In some implementations, semantic similarity is facilitated using a dictionary or “thesaurus” that categorizes words or phrases as having the same or similar meanings.

The encoded training sentences may be used to train a similarity function **606**. Specifically, a computer-generated model may be trained such that a mathematical equation is generated. Typically, the computer-generated model includes a plurality of variables or expressions, where each of the variables or expressions is associated with a different coefficient.

Test sentences **608** may be used to test or refine similarity function **606**. For example, test sentences **608** can include pairs of semantically similar sentences and/or non-semantically similar sentences that were not used to train the similarity function **606**. The pairs of sentences may then be processed by similarity function **606** and resulting scores may be examined to ascertain whether similarity function **606** accurately determined the similarity of the test sentence pairs.

In some implementations, similarity function **606** compares two sentences (e.g., questions) and generates a score indicating a level of similarity between the two sentences. Where a given sentence is compared to a plurality of sentences, a plurality of scores are generated that each represents the similarity between the sentence and a corresponding one of the sentences. Based upon the scores generated by similarity function **606**, it is possible to identify at least one of the sentences that is most similar to the sentence. For example, by ranking the sentences by similarity score, it is possible to identify sentence(s) having the highest (or lowest) similarity score.

FIG. **6B** shows a diagram **650** illustrating the training of a question generation function, in accordance with some implementations. As shown in this example, a set of question generation training data **652** may be used to generate question generation function **654**. For example, training data **652** can include a set of documents and questions that have been generated from the documents. The set of documents can include one or more sentences. For example, the sen-

tences can include one or more interrogative sentences that each has a question mark at the end of the corresponding sentence.

As disclosed herein, questions may be generated by extracting the questions from the set of documents. In addition, questions may be generated by extracting text segments from the set of documents and appending the text segments to generate corresponding questions. Moreover, questions may be generated by converting statements in the set of documents to questions.

For example, input of the of question generation function **654** can include a sentence such as “I ate breakfast in the morning.” Output of question generation function **654** can include an interrogative sentence such as “Did you eat breakfast in the morning?” Thus, in some implementations, a question may be generated from a statement.

Question generation function **654** may include a computer-generated model, as described herein. More particularly, question generation function **654** can be trained to generate one or more questions based upon input text. Specifically, word embedding may be performed for each word in the set of documents such that text in the set of documents is represented numerically. In some implementations, a universal sentence encoder encodes text into high dimensional vectors. For example, each sentence in the set of documents may be represented by a vector representation, where each value in the vector representation represents a corresponding word in the documents.

Test data **656** can include a second set of documents and corresponding questions. Using the second set of documents, it is possible to determine whether question generation function **654** generates the corresponding questions from the second set of documents.

As discussed above, similarity and question generation functions may be trained using training data. However, these examples are merely illustrative. Thus, similarity and question generation functions may be implemented using other mechanisms. In other implementations, similarity and question generation functions may be implemented using corresponding sets of rules that have been statically configured. These rules can be used to implement pattern recognition for identifying similar sentences or generating questions. Artificial intelligence may also be implemented to generate or update rules for identifying similar sentences or generating questions.

FIG. **7** shows an example of a method **700** for implementing a question answer system, in accordance with some implementations. As shown in FIG. **7**, there are two processes that may be performed in parallel. The first process is a question generation process represented by blocks **702-708**, while the second process is an answer retrieval process represented by blocks **710-716**.

In the first process, question answer system generates at least one question using a set of documents stored in at least one data source (**702**). More particularly, the system may parse the set of documents to extract one or more text segments, which may be used to generate the question. Specifically, question generation can include identification of a question in a document and extraction of the question from the document. In addition, question generation can include using information extracted from the set of documents. For example, a question may be generated by appending two or more text segments that have been extracted from one or more documents. As another example, a question may be generated from a statement. Question generation can be performed using a question generation



## 11

function, which can include application of a computer-generated model, a set of rules, pattern recognition, and/or artificial intelligence.

In some implementations, a rule-based method may apply regular expressions to identify questions within the set of documents. For example, question answer system may apply a rule to extract each sentence having a question mark at the end. Where the set of documents includes live chat transcripts, the rule may extract questions transmitted by an agent or a customer in an agent-customer exchange. As another example, question answer system may apply a set of rules to identify questions within one or more fields of a particular type of CRM record, such as a case.

The system then compares the question with a first plurality of questions within a question-answer repository such that a first plurality of similarity scores are generated, where each of the first plurality of similarity scores represents a similarity between the question and a corresponding one of the first plurality of questions (704). As described above, the first plurality of scores may be generated via application of a similarity function, which can include application of a computer-generated model, a set of rules, pattern recognition, and/or artificial intelligence. The system may then rank the first plurality of questions according to the corresponding similarity scores.

The system may determine whether the question is semantically distinct from the first plurality of questions based, at least in part, on the similarity scores (706). Specifically, scores that indicate that the question is semantically distinct would indicate a low degree of semantic similarity. For example, where a high score indicates a high degree of semantic similarity, a low score would indicate a low degree of semantic similarity. In some implementations, one or more pre-defined threshold values may be used to determine whether two different compared questions are semantically similar or semantically distinct. For example, where a score is below a first threshold, this may indicate that the two compared questions are not semantically similar; in other words, where the score is below the first threshold, this may indicate that the two compared questions are semantically distinct. As another example, where a score is above a second threshold, the system may conclude that the corresponding compared questions are semantically similar.

The system may then add the question to the question-answer repository based, at least in part, on a result of determining whether the question is semantically distinct from the first plurality of questions (708). Specifically, where the question is determined to be semantically distinct from the first plurality of questions, the system may add the question to the question-answer repository or intermediary data structure. For example, an intermediary data structure may store system-generated questions that do not have corresponding answers. Alternatively, where the question is determined not to be semantically distinct (e.g., determined to be semantically similar to the first plurality of questions), the question is not added to the question-answer repository or intermediary data structure.

The system may obtain an answer to the question and update the question-answer repository to include the answer such that the answer is associated with the question. The answer may be system generated. For example, the system may generate the answer using the set of documents from which the corresponding question was generated or another set of documents. This may be accomplished using a variety of mechanisms including, but not limited to, an answer generation function, a set of rules, pattern recognition,

## 12

and/or Artificial Intelligence. In some implementations, the system may generate the answer after determining that the question is semantically distinct from those already stored in the question-answer repository.

In some implementations, an answer to the system-generated question or indication thereof may be obtained from a client device. For example, unanswered system-generated questions may be made available for answering by a system administrator. This may be accomplished by querying the question-answer repository or an independent data structure to identify unanswered system-generated questions. A system administrator or another individual may submit an answer to the question via an input device. The system may then update the question-answer repository to include the answer responsive such that the system-generated question is associated with the answer in the question-answer repository.

In the second process, the system obtains a user question received from a client device (710). The user may submit a question via an input device such as a keyboard or microphone. For example, the user may type a question in an input text box rendered via a graphical user interface (GUI).

The system may compare the user question with a second plurality of questions within the question-answer repository such that a second plurality of similarity scores are generated, where each of the second plurality of similarity scores represents a similarity between the user question and a corresponding one of the second plurality of questions (712). By ranking the scores, the system may facilitate the identification of questions that are semantically similar.

The system may then identify one or more of the second plurality of questions that are semantically similar to the user question based, at least in part, on the second plurality of similarity scores (714). For example, the system may identify questions for which corresponding scores exceed a particular threshold value.

For each of the identified one or more of the second plurality of questions, the system may retrieve a corresponding answer from the question-answer repository and provide the corresponding answer (716). In some implementations, a question-answer pair including the answer is provided. The answer or question-answer pair may be provided for presentation via a client device. For example, the answer or question-answer pair may be provided for display via a display device or for presentation via a speaker.

In some implementations, in the event that the system cannot identify one or more of the second plurality of questions that are semantically similar to a user-submitted question (e.g., the user-submitted question is semantically distinct from questions already stored in the question-answer repository), the system may add the user-submitted question to the question-answer repository. For example, the user-submitted question may be stored in a question-answer log file, which is subsequently processed by the system. In this manner, answer retrieval may be integrated with question generation.

Some but not all of the techniques described or referenced herein are implemented using or in conjunction with a social networking system. Social networking systems have become a popular way to facilitate communication among people, any of whom can be recognized as users of a social networking system. One example of a social networking system is Chatter®, provided by salesforce.com, inc. of San Francisco, California salesforce.com, inc. is a provider of social networking services, CRM services and other database management services, any of which can be accessed and used in conjunction with the techniques disclosed herein in some



implementations. In some but not all implementations, these various services can be provided in a cloud computing environment, for example, in the context of a multi-tenant database system. Thus, the disclosed techniques can be implemented without having to install software locally, that is, on computing devices of users interacting with services available through the cloud. While the disclosed implementations are often described with reference to Chatter®, those skilled in the art should understand that the disclosed techniques are neither limited to Chatter® nor to any other services and systems provided by salesforce.com, inc. and can be implemented in the context of various other database systems and/or social networking systems such as Facebook®, LinkedIn®, Twitter®, Google+®, Yammer® and Jive® by way of example only.

Some social networking systems can be implemented in various settings, including organizations. For instance, a social networking system can be implemented to connect users within an enterprise such as a company or business partnership, or a group of users within such an organization. For instance, Chatter® can be used by employee users in a division of a business organization to share data, communicate, and collaborate with each other for various social purposes often involving the business of the organization. In the example of a multi-tenant database system, each organization or group within the organization can be a respective tenant of the system, as described in greater detail below.

In some social networking systems, users can access one or more social network feeds, which include information updates presented as items or entries in the feed. Such a feed item can include a single information update or a collection of individual information updates. A feed item can include various types of data including character-based data, audio data, image data and/or video data. A social network feed can be displayed in a graphical user interface (GUI) on a display device such as the display of a computing device as described below. The information updates can include various social network data from various sources and can be stored in a database system. In some but not all implementations, the disclosed methods, apparatus, systems, and computer program products may be configured or designed for use in a multi-tenant database environment.

In some implementations, a social networking system may allow a user to follow data objects in the form of CRM records such as cases, accounts, or opportunities, in addition to following individual users and groups of users. The “following” of a record stored in a database, as described in greater detail below, allows a user to track the progress of that record when the user is subscribed to the record. Updates to the record, also referred to herein as changes to the record, are one type of information update that can occur and be noted on a social network feed such as a record feed or a news feed of a user subscribed to the record. Examples of record updates include field changes in the record, updates to the status of a record, as well as the creation of the record itself. Some records are publicly accessible, such that any user can follow the record, while other records are private, for which appropriate security clearance/permissions are a prerequisite to a user following the record.

Information updates can include various types of updates, which may or may not be linked with a particular record. For example, information updates can be social media messages submitted by a user or can be otherwise generated in response to user actions or in response to events. Examples of social media messages include: posts, comments, indications of a user’s personal preferences such as “likes” and “dislikes”, updates to a user’s status, uploaded files, and

user-submitted hyperlinks to social network data or other network data such as various documents and/or web pages on the Internet. Posts can include alpha-numeric or other character-based user inputs such as words, phrases, statements, questions, emotional expressions, and/or symbols. Comments generally refer to responses to posts or to other information updates, such as words, phrases, statements, answers, questions, and reactionary emotional expressions and/or symbols. Multimedia data can be included in, linked with, or attached to a post or comment. For example, a post can include textual statements in combination with a JPEG image or animated image. A like or dislike can be submitted in response to a particular post or comment. Examples of uploaded files include presentations, documents, multimedia files, and the like.

Users can follow a record by subscribing to the record, as mentioned above. Users can also follow other entities such as other types of data objects, other users, and groups of users. Feed tracked updates regarding such entities are one type of information update that can be received and included in the user’s news feed. Any number of users can follow a particular entity and thus view information updates pertaining to that entity on the users’ respective news feeds. In some social networks, users may follow each other by establishing connections with each other, sometimes referred to as “friending” one another. By establishing such a connection, one user may be able to see information generated by, generated about, or otherwise associated with another user. For instance, a first user may be able to see information posted by a second user to the second user’s personal social network page. One implementation of such a personal social network page is a user’s profile page, for example, in the form of a web page representing the user’s profile. In one example, when the first user is following the second user, the first user’s news feed can receive a post from the second user submitted to the second user’s profile feed. A user’s profile feed is also referred to herein as the user’s “wall,” which is one example of a social network feed displayed on the user’s profile page.

In some implementations, a social network feed may be specific to a group of users of a social networking system. For instance, a group of users may publish a feed. Members of the group may view and post to this group feed in accordance with a permissions configuration for the feed and the group. Information updates in a group context can also include changes to group status information.

In some implementations, when data such as posts or comments input from one or more users are submitted to a social network feed for a particular user, group, object, or other construct within a social networking system, an email notification or other type of network communication may be transmitted to all users following the user, group, or object in addition to the inclusion of the data as a feed item in one or more feeds, such as a user’s profile feed, a news feed, or a record feed. In some social networking systems, the occurrence of such a notification is limited to the first instance of a published input, which may form part of a larger conversation. For instance, a notification may be transmitted for an initial post, but not for comments on the post. In some other implementations, a separate notification is transmitted for each such information update.

The term “multi-tenant database system” generally refers to those systems in which various elements of hardware and/or software of a database system may be shared by one or more customers. For example, a given application server may simultaneously process requests for a great number of



15

customers, and a given database table may store rows of data such as feed items for a potentially much greater number of customers.

An example of a “user profile” or “user’s profile” is a database object or set of objects configured to store and maintain data about a given user of a social networking system and/or database system. The data can include general information, such as name, title, phone number, a photo, a biographical summary, and a status, e.g., text describing what the user is currently doing. As mentioned below, the data can include social media messages created by other users. Where there are multiple tenants, a user is typically associated with a particular tenant. For example, a user could be a salesperson of a company, which is a tenant of the database system that provides a database service.

The term “record” generally refers to a data entity having fields with values and stored in database system. An example of a record is an instance of a data object created by a user of the database service, for example, in the form of a CRM record about a particular (actual or potential) business relationship or project. The record can have a data structure defined by the database service (a standard object) or defined by a user (custom object). For example, a record can be for a business partner or potential business partner (e.g., a client, vendor, distributor, etc.) of the user, and can include information describing an entire company, subsidiaries, or contacts at the company. As another example, a record can be a project that the user is working on, such as an opportunity (e.g., a possible sale) with an existing partner, or a project that the user is trying to get. In one implementation of a multi-tenant database system, each record for the tenants has a unique identifier stored in a common table. A record has data fields that are defined by the structure of the object (e.g., fields of certain data types and purposes). A record can also have custom fields defined by a user. A field can be another record or include links thereto, thereby providing a parent-child relationship between the records.

The terms “social network feed” and “feed” are used interchangeably herein and generally refer to a combination (e.g., a list) of feed items or entries with various types of information and data. Such feed items can be stored and maintained in one or more database tables, e.g., as rows in the table(s), that can be accessed to retrieve relevant information to be presented as part of a displayed feed. The term “feed item” (or feed element) generally refers to an item of information, which can be presented in the feed such as a post submitted by a user. Feed items of information about a user can be presented in a user’s profile feed of the database, while feed items of information about a record can be presented in a record feed in the database, by way of example. A profile feed and a record feed are examples of different types of social network feeds. A second user following a first user and a record can receive the feed items associated with the first user and the record for display in the second user’s news feed, which is another type of social network feed. In some implementations, the feed items from any number of followed users and records can be combined into a single social network feed of a particular user.

As examples, a feed item can be a social media message, such as a user-generated post of text data, and a feed tracked update to a record or profile, such as a change to a field of the record. Feed tracked updates are described in greater detail below. A feed can be a combination of social media messages and feed tracked updates. Social media messages include text created by a user, and may include other data as well. Examples of social media messages include posts, user status updates, and comments. Social media messages can

16

be created for a user’s profile or for a record. Posts can be created by various users, potentially any user, although some restrictions can be applied. As an example, posts can be made to a wall section of a user’s profile page (which can include a number of recent posts) or a section of a record that includes multiple posts. The posts can be organized in chronological order when displayed in a GUI, for instance, on the user’s profile page, as part of the user’s profile feed. In contrast to a post, a user status update changes a status of a user and can be made by that user or an administrator. A record can also have a status, the update of which can be provided by an owner of the record or other users having suitable write access permissions to the record. The owner can be a single user, multiple users, or a group.

In some implementations, a comment can be made on any feed item. In some implementations, comments are organized as a list explicitly tied to a particular feed tracked update, post, or status update. In some implementations, comments may not be listed in the first layer (in a hierarchical sense) of feed items, but listed as a second layer branching from a particular first layer feed item.

A “feed tracked update,” also referred to herein as a “feed update,” is one type of information update and generally refers to data representing an event. A feed tracked update can include text generated by the database system in response to the event, to be provided as one or more feed items for possible inclusion in one or more feeds. In one implementation, the data can initially be stored, and then the database system can later use the data to create text for describing the event. Both the data and/or the text can be a feed tracked update, as used herein. In various implementations, an event can be an update of a record and/or can be triggered by a specific action by a user. Which actions trigger an event can be configurable. Which events have feed tracked updates created and which feed updates are sent to which users can also be configurable. Social media messages and other types of feed updates can be stored as a field or child object of the record. For example, the feed can be stored as a child object of the record.

A “group” is generally a collection of users. In some implementations, the group may be defined as users with a same or similar attribute, or by membership. In some implementations, a “group feed”, also referred to herein as a “group news feed”, includes one or more feed items about any user in the group. In some implementations, the group feed also includes information updates and other feed items that are about the group as a whole, the group’s purpose, the group’s description, and group records and other objects stored in association with the group.

Threads of information updates including group record updates and social media messages, such as posts, comments, likes, etc., can define group conversations and change over time.

An “entity feed” or “record feed” generally refers to a feed of feed items about a particular record in the database. Such feed items can include feed tracked updates about changes to the record and posts made by users about the record. An entity feed can be composed of any type of feed item. Such a feed can be displayed on a page such as a web page associated with the record, e.g., a home page of the record. As used herein, a “profile feed” or “user’s profile feed” generally refers to a feed of feed items about a particular user. In one example, the feed items for a profile feed include posts and comments that other users make about or send to the particular user, and status updates made by the particular user. Such a profile feed can be displayed on a page associated with the particular user. In another



17

example, feed items in a profile feed could include posts made by the particular user and feed tracked updates initiated based on actions of the particular user.

Some non-limiting examples of systems, apparatus, and methods are described below for implementing database systems and enterprise level social networking systems in conjunction with the disclosed techniques. Such implementations can provide more efficient use of a database system. For instance, a user of a database system may not easily know when important information in the database has changed, e.g., about a project or client. Such implementations can provide feed tracked updates about such changes and other events, thereby keeping users informed.

FIG. 8A shows a block diagram of an example of an environment 10 in which an on-demand database service exists and can be used in accordance with some implementations. Environment 10 may include user systems 12, network 14, database system 16, processor system 17, application platform 18, network interface 20, tenant data storage 22, system data storage 24, program code 26, and process space 28. In other implementations, environment 10 may not have all of these components and/or may have other components instead of, or in addition to, those listed above.

A user system 12 may be implemented as any computing device(s) or other data processing apparatus such as a machine or system used by a user to access a database system 16. For example, any of user systems 12 can be a handheld and/or portable computing device such as a mobile phone, a smartphone, a laptop computer, or a tablet. Other examples of a user system include computing devices such as a work station and/or a network of computing devices. As illustrated in FIG. 8A (and in more detail in FIG. 8B) user systems 12 might interact via a network 14 with an on-demand database service, which is implemented in the example of FIG. 8A as database system 16.

An on-demand database service, implemented using system 16 by way of example, is a service that is made available to users who do not need to necessarily be concerned with building and/or maintaining the database system. Instead, the database system may be available for their use when the users need the database system, i.e., on the demand of the users. Some on-demand database services may store information from one or more tenants into tables of a common database image to form a multi-tenant database system (MTS). A database image may include one or more database objects. A relational database management system (RDBMS) or the equivalent may execute storage and retrieval of information against the database object(s). Application platform 18 may be a framework that allows the applications of system 16 to run, such as the hardware and/or software, e.g., the operating system. In some implementations, application platform 18 enables creation, managing and executing one or more applications developed by the provider of the on-demand database service, users accessing the on-demand database service via user systems 12, or third party application developers accessing the on-demand database service via user systems 12.

The users of user systems 12 may differ in their respective capacities, and the capacity of a particular user system 12 might be entirely determined by permissions (permission levels) for the current user. For example, when a salesperson is using a particular user system 12 to interact with system 16, the user system has the capacities allotted to that salesperson. However, while an administrator is using that user system to interact with system 16, that user system has the capacities allotted to that administrator. In systems with a hierarchical role model, users at one permission level may

18

have access to applications, data, and database information accessible by a lower permission level user, but may not have access to certain applications, database information, and data accessible by a user at a higher permission level. Thus, different users will have different capabilities with regard to accessing and modifying application and database information, depending on a user's security or permission level, also called authorization.

Network 14 is any network or combination of networks of devices that communicate with one another. For example, network 14 can be any one or any combination of a LAN (local area network), WAN (wide area network), telephone network, wireless network, point-to-point network, star network, token ring network, hub network, or other appropriate configuration. Network 14 can include a TCP/IP (Transfer Control Protocol and Internet Protocol) network, such as the global internetwork of networks often referred to as the Internet. The Internet will be used in many of the examples herein. However, it should be understood that the networks that the present implementations might use are not so limited.

User systems 12 might communicate with system 16 using TCP/IP and, at a higher network level, use other common Internet protocols to communicate, such as HTTP, FTP, AFS, WAP, etc. In an example where HTTP is used, user system 12 might include an HTTP client commonly referred to as a "browser" for sending and receiving HTTP signals to and from an HTTP server at system 16. Such an HTTP server might be implemented as the sole network interface 20 between system 16 and network 14, but other techniques might be used as well or instead. In some implementations, the network interface 20 between system 16 and network 14 includes load sharing functionality, such as round-robin HTTP request distributors to balance loads and distribute incoming HTTP requests evenly over a plurality of servers. At least for users accessing system 16, each of the plurality of servers has access to the MTS' data; however, other alternative configurations may be used instead.

In one implementation, system 16, shown in FIG. 8A, implements a web-based CRM system. For example, in one implementation, system 16 includes application servers configured to implement and execute CRM software applications as well as provide related data, code, forms, web pages and other information to and from user systems 12 and to store to, and retrieve from, a database system related data, objects, and Webpage content. With a multi-tenant system, data for multiple tenants may be stored in the same physical database object in tenant data storage 22, however, tenant data typically is arranged in the storage medium(s) of tenant data storage 22 so that data of one tenant is kept logically separate from that of other tenants so that one tenant does not have access to another tenant's data, unless such data is expressly shared. In certain implementations, system 16 implements applications other than, or in addition to, a CRM application. For example, system 16 may provide tenant access to multiple hosted (standard and custom) applications, including a CRM application. User (or third party developer) applications, which may or may not include CRM, may be supported by the application platform 18, which manages creation, storage of the applications into one or more database objects and executing of the applications in a virtual machine in the process space of the system 16.

One arrangement for elements of system 16 is shown in FIGS. 7A and 7B, including a network interface 20, application platform 18, tenant data storage 22 for tenant data 23, system data storage 24 for system data 25 accessible to



19

system 16 and possibly multiple tenants, program code 26 for implementing various functions of system 16, and a process space 28 for executing MTS system processes and tenant-specific processes, such as running applications as part of an application hosting service. Additional processes that may execute on system 16 include database indexing processes.

Several elements in the system shown in FIG. 8A include conventional, well-known elements that are explained only briefly here. For example, each user system 12 could include a desktop personal computer, workstation, laptop, PDA, cell phone, or any wireless access protocol (WAP) enabled device or any other computing device capable of interfacing directly or indirectly to the Internet or other network connection. The term “computing device” is also referred to herein simply as a “computer”. User system 12 typically runs an HTTP client, e.g., a browsing program, such as Microsoft’s Internet Explorer browser, Netscape’s Navigator browser, Opera’s browser, or a WAP-enabled browser in the case of a cell phone, PDA or other wireless device, or the like, allowing a user (e.g., subscriber of the multi-tenant database system) of user system 12 to access, process and view information, pages and applications available to it from system 16 over network 14. Each user system 12 also typically includes one or more user input devices, such as a keyboard, a mouse, trackball, touch pad, touch screen, pen or the like, for interacting with a GUI provided by the browser on a display (e.g., a monitor screen, LCD display, OLED display, etc.)

of the computing device in conjunction with pages, forms, applications and other information provided by system 16 or other systems or servers. Thus, “display device” as used herein can refer to a display of a computer system such as a monitor or touch-screen display, and can refer to any computing device having display capabilities such as a desktop computer, laptop, tablet, smartphone, a television set-top box, or wearable device such Google Glass® or other human body-mounted display apparatus. For example, the display device can be used to access data and applications hosted by system 16, and to perform searches on stored data, and otherwise allow a user to interact with various GUI pages that may be presented to a user. As discussed above, implementations are suitable for use with the Internet, although other networks can be used instead of or in addition to the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

According to one implementation, each user system 12 and all of its components are operator configurable using applications, such as a browser, including computer code run using a central processing unit such as an Intel Pentium® processor or the like. Similarly, system 16 (and additional instances of an MTS, where more than one is present) and all of its components might be operator configurable using application(s) including computer code to run using processor system 17, which may be implemented to include a central processing unit, which may include an Intel Pentium® processor or the like, and/or multiple processor units. Non-transitory computer-readable media can have instructions stored thereon/in, that can be executed by or used to program a computing device to perform any of the methods of the implementations described herein. Computer program code 26 implementing instructions for operating and configuring system 16 to intercommunicate and to process web pages, applications and other data and media content as described herein is preferably downloadable and stored on a hard disk, but the entire program code, or portions thereof,

20

may also be stored in any other volatile or non-volatile memory medium or device as is well known, such as a ROM or RAM, or provided on any media capable of storing program code, such as any type of rotating media including floppy disks, optical discs, digital versatile disk (DVD), compact disk (CD), microdrive, and magneto-optical disks, and magnetic or optical cards, nanosystems (including molecular memory ICs), or any other type of computer-readable medium or device suitable for storing instructions and/or data. Additionally, the entire program code, or portions thereof, may be transmitted and downloaded from a software source over a transmission medium, e.g., over the Internet, or from another server, as is well known, or transmitted over any other conventional network connection as is well known (e.g., extranet, VPN, LAN, etc.) using any communication medium and protocols (e.g., TCP/IP, HTTP, HTTPS, Ethernet, etc.) as are well known. It will also be appreciated that computer code for the disclosed implementations can be realized in any programming language that can be executed on a client system and/or server or server system such as, for example, C, C++, HTML, any other markup language, Java™, JavaScript, ActiveX, any other scripting language, such as VBScript, and many other programming languages as are well known may be used. (Java™ is a trademark of Sun Microsystems, Inc.).

According to some implementations, each system 16 is configured to provide web pages, forms, applications, data and media content to user (client) systems 12 to support the access by user systems 12 as tenants of system 16. As such, system 16 provides security mechanisms to keep each tenant’s data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (e.g., in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (e.g., one or more servers located in city A and one or more servers located in city B). As used herein, each MTS could include one or more logically and/or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term “server” is meant to refer to one type of computing device such as a system including processing hardware and process space(s), an associated storage medium such as a memory device or database, and, in some instances, a database application (e.g., OODBMS or RDBMS) as is well known in the art. It should also be understood that “server system” and “server” are often used interchangeably herein. Similarly, the database objects described herein can be implemented as single databases, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, etc., and might include a distributed database or storage network and associated processing intelligence.

FIG. 8B shows a block diagram of an example of some implementations of elements of FIG. 8A and various possible interconnections between these elements. That is, FIG. 8B also illustrates environment 10. However, in FIG. 8B elements of system 16 and various interconnections in some implementations are further illustrated. FIG. 8B shows that user system 12 may include processor system 12A, memory system 12B, input system 12C, and output system 12D. FIG. 8B shows network 14 and system 16. FIG. 8B also shows that system 16 may include tenant data storage 22, tenant data 23, system data storage 24, system data 25, User Interface (UI) 30, Application Program Interface (API) 32, PL/SQL 34, save routines 36, application setup mechanism 38, application servers 50<sub>1</sub>-50<sub>N</sub>, system process space 52, tenant process spaces 54, tenant management process



## 21

space 60, tenant storage space 62, user storage 64, and application metadata 66. In other implementations, environment 10 may not have the same elements as those listed above and/or may have other elements instead of, or in addition to, those listed above.

User system 12, network 14, system 16, tenant data storage 22, and system data storage 24 were discussed above in FIG. 8A. Regarding user system 12, processor system 12A may be any combination of one or more processors. Memory system 12B may be any combination of one or more memory devices, short term, and/or long term memory. Input system 12C may be any combination of input devices, such as one or more keyboards, mice, trackballs, scanners, cameras, and/or interfaces to networks. Output system 12D may be any combination of output devices, such as one or more monitors, printers, and/or interfaces to networks. As shown by FIG. 8B, system 16 may include a network interface 20 (of FIG. 8A) implemented as a set of application servers 50, an application platform 18, tenant data storage 22, and system data storage 24. Also shown is system process space 52, including individual tenant process spaces 54 and a tenant management process space 60. Each application server 50 may be configured to communicate with tenant data storage 22 and the tenant data 23 therein, and system data storage 24 and the system data 25 therein to serve requests of user systems 12. The tenant data 23 might be divided into individual tenant storage spaces 62, which can be either a physical arrangement and/or a logical arrangement of data. Within each tenant storage space 62, user storage 64 and application metadata 66 might be similarly allocated for each user. For example, a copy of a user's most recently used (MRU) items might be stored to user storage 64. Similarly, a copy of MRU items for an entire organization that is a tenant might be stored to tenant storage space 62. A UI 30 provides a user interface and an API 32 provides an application programmer interface to system 16 resident processes to users and/or developers at user systems 12. The tenant data and the system data may be stored in various databases, such as one or more Oracle® databases.

Application platform 18 includes an application setup mechanism 38 that supports application developers' creation and management of applications, which may be saved as metadata into tenant data storage 22 by save routines 36 for execution by subscribers as one or more tenant process spaces 54 managed by tenant management process 60 for example. Invocations to such applications may be coded using PL/SOQL 34 that provides a programming language style interface extension to API 32. A detailed description of some PL/SOQL language implementations is discussed in commonly assigned U.S. Pat. No. 7,730,478, titled METHOD AND SYSTEM FOR ALLOWING ACCESS TO DEVELOPED APPLICATIONS VIA A MULTI-TENANT ON-DEMAND DATABASE SERVICE, by Craig Weissman, issued on Jun. 1, 2010, and hereby incorporated by reference in its entirety and for all purposes. Invocations to applications may be detected by one or more system processes, which manage retrieving application metadata 66 for the subscriber making the invocation and executing the metadata as an application in a virtual machine. Each application server 50 may be communicably coupled to database systems, e.g., having access to system data 25 and tenant data 23, via a different network connection. For example, one application server 50<sub>1</sub> might be coupled via the network 14 (e.g., the Internet), another application server 50<sub>N-1</sub> might be coupled via a direct network link, and another application server 50<sub>N</sub> might be coupled by yet a different network connection. Transfer Control Protocol and

## 22

Internet Protocol (TCP/IP) are typical protocols for communicating between application servers 50 and the database system. However, it will be apparent to one skilled in the art that other transport protocols may be used to optimize the system depending on the network interconnect used.

In certain implementations, each application server 50 is configured to handle requests for any user associated with any organization that is a tenant. Because it is desirable to be able to add and remove application servers from the server pool at any time for any reason, there is preferably no server affinity for a user and/or organization to a specific application server 50. In one implementation, therefore, an interface system implementing a load balancing function (e.g., an F5 Big-IP load balancer) is communicably coupled between the application servers 50 and the user systems 12 to distribute requests to the application servers 50. In one implementation, the load balancer uses a least connections algorithm to route user requests to the application servers 50. Other examples of load balancing algorithms, such as round robin and observed response time, also can be used. For example, in certain implementations, three consecutive requests from the same user could hit three different application servers 50, and three requests from different users could hit the same application server 50. In this manner, by way of example, system 16 is multi-tenant, wherein system 16 handles storage of, and access to, different objects, data and applications across disparate users and organizations.

As an example of storage, one tenant might be a company that employs a sales force where each salesperson uses system 16 to manage their sales process. Thus, a user might maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, etc., all applicable to that user's personal sales process (e.g., in tenant data storage 22). In an example of a MTS arrangement, since all of the data and the applications to access, view, modify, report, transmit, calculate, etc., can be maintained and accessed by a user system having nothing more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, if a salesperson is visiting a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates as to that customer while waiting for the customer to arrive in the lobby.

While each user's data might be separate from other users' data regardless of the employers of each user, some data might be organization-wide data shared or accessible by a plurality of users or all of the users for a given organization that is a tenant. Thus, there might be some data structures managed by system 16 that are allocated at the tenant level while other data structures might be managed at the user level. Because an MTS might support multiple tenants including possible competitors, the MTS should have security protocols that keep data, applications, and application use separate. Also, because many tenants may opt for access to an MTS rather than maintain their own system, redundancy, up-time, and backup are additional functions that may be implemented in the MTS. In addition to user-specific data and tenant-specific data, system 16 might also maintain system level data usable by multiple tenants or other data. Such system level data might include industry reports, news, postings, and the like that are sharable among tenants.

In certain implementations, user systems 12 (which may be client systems) communicate with application servers 50 to request and update system-level and tenant-level data from system 16 that may involve sending one or more queries to tenant data storage 22 and/or system data storage 24. System 16 (e.g., an application server 50 in system 16)



automatically generates one or more SQL statements (e.g., one or more SQL queries) that are designed to access the desired information. System data storage **24** may generate query plans to access the requested data from the database.

Each database can generally be viewed as a collection of objects, such as a set of logical tables, containing data fitted into predefined categories. A “table” is one representation of a data object, and may be used herein to simplify the conceptual description of objects and custom objects according to some implementations. It should be understood that “table” and “object” may be used interchangeably herein. Each table generally contains one or more data categories logically arranged as columns or fields in a viewable schema. Each row or record of a table contains an instance of data for each category defined by the fields. For example, a CRM database may include a table that describes a customer with fields for basic contact information such as name, address, phone number, fax number, etc. Another table might describe a purchase order, including fields for information such as customer, product, sale price, date, etc. In some multi-tenant database systems, standard entity tables might be provided for use by all tenants. For CRM database applications, such standard entities might include tables for case, account, contact, lead, and opportunity data objects, each containing pre-defined fields. It should be understood that the word “entity” may also be used interchangeably herein with “object” and “table”.

In some multi-tenant database systems, tenants may be allowed to create and store custom objects, or they may be allowed to customize standard entities or objects, for example by creating custom fields for standard objects, including custom index fields. Commonly assigned U.S. Pat. No. 7,779,039, titled CUSTOM ENTITIES AND FIELDS IN A MULTI-TENANT DATABASE SYSTEM, by Weissman et al., issued on Aug. 17, 2010, and hereby incorporated by reference in its entirety and for all purposes, teaches systems and methods for creating custom objects as well as customizing standard objects in a multi-tenant database system. In certain implementations, for example, all custom entity data rows are stored in a single multi-tenant physical table, which may contain multiple logical tables per organization. It is transparent to customers that their multiple “tables” are in fact stored in one large table or that their data may be stored in the same table as the data of other customers.

FIG. 9A shows a system diagram of an example of architectural components of an on-demand database service environment **900**, in accordance with some implementations. A client machine located in the cloud **904**, generally referring to one or more networks in combination, as described herein, may communicate with the on-demand database service environment via one or more edge routers **908** and **912**. A client machine can be any of the examples of user systems **12** described above. The edge routers may communicate with one or more core switches **920** and **924** via firewall **916**. The core switches may communicate with a load balancer **928**, which may distribute server load over different pods, such as the pods **940** and **944**. The pods **940** and **944**, which may each include one or more servers and/or other computing resources, may perform data processing and other operations used to provide on-demand services. Communication with the pods may be conducted via pod switches **932** and **936**. Components of the on-demand database service environment may communicate with a database storage **956** via a database firewall **948** and a database switch **952**.

As shown in FIGS. 8A and 8B, accessing an on-demand database service environment may involve communications transmitted among a variety of different hardware and/or software components. Further, the on-demand database service environment **900** is a simplified representation of an actual on-demand database service environment. For example, while only one or two devices of each type are shown in FIGS. 8A and 8B, some implementations of an on-demand database service environment may include anywhere from one to many devices of each type. Also, the on-demand database service environment need not include each device shown in FIGS. 8A and 8B, or may include additional devices not shown in FIGS. 8A and 8B.

Moreover, one or more of the devices in the on-demand database service environment **900** may be implemented on the same physical device or on different hardware. Some devices may be implemented using hardware or a combination of hardware and software. Thus, terms such as “data processing apparatus,” “machine,” “server” and “device” as used herein are not limited to a single hardware device, but rather include any hardware and software configured to provide the described functionality.

The cloud **904** is intended to refer to a data network or combination of data networks, often including the Internet. Client machines located in the cloud **904** may communicate with the on-demand database service environment to access services provided by the on-demand database service environment. For example, client machines may access the on-demand database service environment to retrieve, store, edit, and/or process information.

In some implementations, the edge routers **908** and **912** route packets between the cloud **904** and other components of the on-demand database service environment **900**. The edge routers **908** and **912** may employ the Border Gateway Protocol (BGP). The BGP is the core routing protocol of the Internet. The edge routers **908** and **912** may maintain a table of IP networks or ‘prefixes’, which designate network reachability among autonomous systems on the Internet.

In one or more implementations, the firewall **916** may protect the inner components of the on-demand database service environment **900** from Internet traffic. The firewall **916** may block, permit, or deny access to the inner components of the on-demand database service environment **900** based upon a set of rules and other criteria. The firewall **916** may act as one or more of a packet filter, an application gateway, a stateful filter, a proxy server, or any other type of firewall.

In some implementations, the core switches **920** and **924** are high-capacity switches that transfer packets within the on-demand database service environment **900**. The core switches **920** and **924** may be configured as network bridges that quickly route data between different components within the on-demand database service environment. In some implementations, the use of two or more core switches **920** and **924** may provide redundancy and/or reduced latency.

In some implementations, the pods **940** and **944** may perform the core data processing and service functions provided by the on-demand database service environment. Each pod may include various types of hardware and/or software computing resources. An example of the pod architecture is discussed in greater detail with reference to FIG. 9B.

In some implementations, communication between the pods **940** and **944** may be conducted via the pod switches **932** and **936**. The pod switches **932** and **936** may facilitate communication between the pods **940** and **944** and client machines located in the cloud **904**, for example via core



## 25

switches **920** and **924**. Also, the pod switches **932** and **936** may facilitate communication between the pods **940** and **944** and the database storage **956**.

In some implementations, the load balancer **928** may distribute workload between the pods **940** and **944**. Balancing the on-demand service requests between the pods may assist in improving the use of resources, increasing throughput, reducing response times, and/or reducing overhead. The load balancer **928** may include multilayer switches to analyze and forward traffic.

In some implementations, access to the database storage **956** may be guarded by a database firewall **948**. The database firewall **948** may act as a computer application firewall operating at the database application layer of a protocol stack. The database firewall **948** may protect the database storage **956** from application attacks such as structure query language (SQL) injection, database rootkits, and unauthorized information disclosure.

In some implementations, the database firewall **948** may include a host using one or more forms of reverse proxy services to proxy traffic before passing it to a gateway router. The database firewall **948** may inspect the contents of database traffic and block certain content or database requests. The database firewall **948** may work on the SQL application level atop the TCP/IP stack, managing applications' connection to the database or SQL management interfaces as well as intercepting and enforcing packets traveling to or from a database network or application interface.

In some implementations, communication with the database storage **956** may be conducted via the database switch **952**. The multi-tenant database storage **956** may include more than one hardware and/or software components for handling database queries. Accordingly, the database switch **952** may direct database queries transmitted by other components of the on-demand database service environment (e.g., the pods **940** and **944**) to the correct components within the database storage **956**.

In some implementations, the database storage **956** is an on-demand database system shared by many different organizations. The on-demand database service may employ a multi-tenant approach, a virtualized approach, or any other type of database approach. On-demand database services are discussed in greater detail with reference to FIGS. **9A** and **9B**.

FIG. **9B** shows a system diagram further illustrating an example of architectural components of an on-demand database service environment, in accordance with some implementations. The pod **944** may be used to render services to a user of the on-demand database service environment **900**. In some implementations, each pod may include a variety of servers and/or other systems. The pod **944** includes one or more content batch servers **964**, content search servers **968**, query servers **982**, file servers **986**, access control system (ACS) servers **980**, batch servers **984**, and app servers **988**. Also, the pod **944** includes database instances **990**, quick file systems (QFS) **992**, and indexers **994**. In one or more implementations, some or all communication between the servers in the pod **944** may be transmitted via the switch **936**.

The content batch servers **964** may handle requests internal to the pod. These requests may be long-running and/or not tied to a particular customer. For example, the content batch servers **964** may handle requests related to log mining, cleanup work, and maintenance tasks.

The content search servers **968** may provide query and indexer functions. For example, the functions provided by

## 26

the content search servers **968** may allow users to search through content stored in the on-demand database service environment.

The file servers **986** may manage requests for information stored in the file storage **998**. The file storage **998** may store information such as documents, images, and basic large objects (BLOBs). By managing requests for information using the file servers **986**, the image footprint on the database may be reduced.

The query servers **982** may be used to retrieve information from one or more file systems. For example, the query system **982** may receive requests for information from the app servers **988** and then transmit information queries to the NFS **996** located outside the pod.

The pod **944** may share a database instance **990** configured as a multi-tenant environment in which different organizations share access to the same database. Additionally, services rendered by the pod **944** may call upon various hardware and/or software resources. In some implementations, the ACS servers **980** may control access to data, hardware resources, or software resources.

In some implementations, the batch servers **984** may process batch jobs, which are used to run tasks at specified times. Thus, the batch servers **984** may transmit instructions to other servers, such as the app servers **988**, to trigger the batch jobs.

In some implementations, the QFS **992** may be an open source file system available from Sun Microsystems® of Santa Clara, California. The QFS may serve as a rapid-access file system for storing and accessing information available within the pod **944**. The QFS **992** may support some volume management capabilities, allowing many disks to be grouped together into a file system. File system metadata can be kept on a separate set of disks, which may be useful for streaming applications where long disk seeks cannot be tolerated. Thus, the QFS system may communicate with one or more content search servers **968** and/or indexers **994** to identify, retrieve, move, and/or update data stored in the network file systems **996** and/or other storage systems.

In some implementations, one or more query servers **982** may communicate with the NFS **996** to retrieve and/or update information stored outside of the pod **944**. The NFS **996** may allow servers located in the pod **944** to access information to access files over a network in a manner similar to how local storage is accessed.

In some implementations, queries from the query servers **922** may be transmitted to the NFS **996** via the load balancer **928**, which may distribute resource requests over various resources available in the on-demand database service environment. The NFS **996** may also communicate with the QFS **992** to update the information stored on the NFS **996** and/or to provide information to the QFS **992** for use by servers located within the pod **944**.

In some implementations, the pod may include one or more database instances **990**. The database instance **990** may transmit information to the QFS **992**. When information is transmitted to the QFS, it may be available for use by servers within the pod **944** without using an additional database call.

In some implementations, database information may be transmitted to the indexer **994**. Indexer **994** may provide an index of information available in the database **990** and/or QFS **992**. The index information may be provided to file servers **986** and/or the QFS **992**.

In some implementations, one or more application servers or other servers described above with reference to FIGS. **7A** and **7B** include a hardware and/or software framework



configurable to execute procedures using programs, routines, scripts, etc. Thus, in some implementations, one or more of application servers **50<sub>1</sub>-50<sub>N</sub>** of FIG. **8B** can be configured to initiate performance of one or more of the operations described above by instructing another computing device to perform an operation. In some implementations, one or more application servers **50<sub>1</sub>-50<sub>N</sub>** carry out, either partially or entirely, one or more of the disclosed operations. In some implementations, app servers **988** of FIG. **9B** support the construction of applications provided by the on-demand database service environment **900** via the pod **944**. Thus, an app server **988** may include a hardware and/or software framework configurable to execute procedures to partially or entirely carry out or instruct another computing device to carry out one or more operations disclosed herein. In alternative implementations, two or more app servers **988** may cooperate to perform or cause performance of such operations. Any of the databases and other storage facilities described above with reference to FIGS. **7A**, **7B**, **8A** and **8B** can be configured to store lists, articles, documents, records, files, and other objects for implementing the operations described above. For instance, lists of available communication channels associated with share actions for sharing a type of data item can be maintained in tenant data storage **22** and/or system data storage **24** of FIGS. **7A** and **7B**. By the same token, lists of default or designated channels for particular share actions can be maintained in storage **22** and/or storage **24**. In some other implementations, rather than storing one or more lists, articles, documents, records, and/or files, the databases and other storage facilities described above can store pointers to the lists, articles, documents, records, and/or files, which may instead be stored in other repositories external to the systems and environments described above with reference to FIGS. **7A**, **7B**, **8A** and **8B**.

While some of the disclosed implementations may be described with reference to a system having an application server providing a front end for an on-demand database service capable of supporting multiple tenants, the disclosed implementations are not limited to multi-tenant databases nor deployment on application servers. Some implementations may be practiced using various database architectures such as ORACLE®, DB2® by IBM and the like without departing from the scope of the implementations claimed.

It should be understood that some of the disclosed implementations can be embodied in the form of control logic using hardware and/or computer software in a modular or integrated manner. Other ways and/or methods are possible using hardware and a combination of hardware and software.

Any of the disclosed implementations may be embodied in various types of hardware, software, firmware, and combinations thereof. For example, some techniques disclosed herein may be implemented, at least in part, by computer-readable media that include program instructions, state information, etc., for performing various services and operations described herein. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher-level code that may be executed by a computing device such as a server or other data processing apparatus using an interpreter. Examples of computer-readable media include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as flash memory, compact disk (CD) or digital versatile disk (DVD); magneto-optical media; and hardware devices specially configured to store program instructions, such as read-only memory (ROM)

devices and random access memory (RAM) devices. A computer-readable medium may be any combination of such storage devices.

Any of the operations and techniques described in this application may be implemented as software code to be executed by a processor using any suitable computer language such as, for example, Java, C++ or Perl using, for example, object-oriented techniques. The software code may be stored as a series of instructions or commands on a computer-readable medium. Computer-readable media encoded with the software/program code may be packaged with a compatible device or provided separately from other devices (e.g., via Internet download). Any such computer-readable medium may reside on or within a single computing device or an entire computer system, and may be among other computer-readable media within a system or network. A computer system or computing device may include a monitor, printer, or other suitable display for providing any of the results mentioned herein to a user.

While various implementations have been described herein, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present application should not be limited by any of the implementations described herein, but should be defined only in accordance with the following and later-submitted claims and their equivalents.

What is claimed is:

1. A system comprising:

a database system implemented using a server system, the database system configurable to cause:

generating a question using a set of documents stored in at least one data source by at least:

identifying a first one or more of the documents as being of a designated type of customer record,

identifying a second one or more of the documents including a transcript of a chat between an agent and a customer associated with the customer record, and processing both the identified first one or more documents and the identified second one or more documents to identify one or more questions stored in one or more designated fields of one or more of: the identified first one or more documents, or the identified second one or more documents;

comparing the generated question with a first plurality of questions within a question-answer repository such that a first plurality of similarity scores are generated, each of the first plurality of similarity scores representing a similarity between the generated question and a corresponding one of the first plurality of questions;

determining whether the generated question is semantically distinct from the first plurality of questions based, at least in part, on the similarity scores;

adding the generated question to the question-answer repository based, at least in part, on having determined that the generated question is semantically distinct from the first plurality of questions; and

updating the set of documents to include the generated question.

2. The system as recited in claim 1, the database system further configurable to cause:

parsing the set of documents to extract one or more text segments;

wherein generating the question is performed using the extracted text segments.

3. The system as recited in claim 1, the database system further configurable to cause:



29

training a question generation model using a set of question generation training data;  
 wherein generating the question is performed using the question generation model.

4. The system as recited in claim 1, the database system further configurable to cause:

- obtaining a user question received from a client device;
- comparing the user question with a second plurality of questions within the question-answer repository such that a second plurality of similarity scores are generated, each of the second plurality of similarity scores representing a similarity between the user question and a corresponding one of the second plurality of questions;
- determining whether the generated question is semantically distinct from the second plurality of questions based, at least in part, on the second plurality of similarity scores; and
- adding the user question to the question-answer repository based, at least in part, on a result of determining whether the user question is semantically distinct from the second plurality of questions.

5. The system as recited in claim 4, the database further configurable to cause:

- training a similarity function using a set of similarity training data;
- wherein comparing the generated question with the first plurality of questions within the question-answer repository and comparing the user question with the second plurality of questions within the question-answer repository are performed using the similarity function.

6. The system as recited in claim 1, the database system further configurable to cause:

- obtaining an answer to the question; and
- updating the question-answer repository to include the answer such that the answer is associated with the question.

7. The system as recited in claim 6, wherein obtaining an answer to the question comprises:

- obtaining an indication of the answer from a client device;
- wherein updating the question-answer repository to include the answer is performed responsive to obtaining the indication of the answer from the client device.

8. The system as recited in claim 6, wherein obtaining an answer to the question comprises automatically obtaining, for the question, a corresponding answer from the set of documents or another set of documents.

9. A method, comprising:

- generating a question using a set of documents stored in at least one data source by at least:
  - identifying a first one or more of the documents as being of a designated type of customer record,
  - identifying a second one or more of the documents including a transcript of a chat between an agent and a customer associated with the customer record, and
  - processing both the identified first one or more documents and the identified second one or more documents to identify one or more questions stored in one or more designated fields of one or more of: the identified first one or more documents, or the identified second one or more documents;
- comparing the generated question with a first plurality of questions within a question-answer repository such that a first plurality of similarity scores are generated, each of the first plurality of similarity scores representing a

30

- similarity between the generated question and a corresponding one of the first plurality of questions;
- determining whether the generated question is semantically distinct from the first plurality of questions based, at least in part, on the similarity scores;
- adding the generated question to the question-answer repository based, at least in part, on having determined that the generated question is semantically distinct from the first plurality of questions; and
- updating the set of documents to include the generated question.

10. The method as recited in claim 9, further comprising: parsing the set of documents to extract one or more text segments;

wherein generating the question is performed using the extracted text segments.

11. The method as recited in claim 9, further comprising: training a question generation model using a set of question generation training data;

wherein generating the question is performed using the question generation model.

12. The method as recited in claim 9, further comprising: obtaining a user question received from a client device;

- comparing the user question with a second plurality of questions within the question-answer repository such that a second plurality of similarity scores are generated, each of the second plurality of similarity scores representing a similarity between the user question and a corresponding one of the second plurality of questions;
- determining whether the generated question is semantically distinct from the second plurality of questions based, at least in part, on the second plurality of similarity scores; and
- adding the user question to the question-answer repository based, at least in part, on a result of determining whether the user question is semantically distinct from the second plurality of questions.

13. The method as recited in claim 12, further comprising: training a similarity function using a set of similarity training data;

wherein comparing the generated question with the first plurality of questions within the question-answer repository and comparing the user question with the second plurality of questions within the question-answer repository are performed using the similarity function.

14. The method as recited in claim 9, further comprising: obtaining an answer to the question; and

- updating the question-answer repository to include the answer such that the answer is associated with the question.

15. A computer program product comprising computer-readable program code capable of being executed by one or more processors when retrieved from a non-transitory computer-readable medium, the program code comprising computer-readable instructions configurable to cause:

- generating a question using a set of documents stored in at least one data source by at least:
  - identifying a first one or more of the documents as being of a designated type of customer record,
  - identifying a second one or more of the documents including a transcript of a chat between an agent and a customer associated with the customer record, and
  - processing both the identified first one or more documents and the identified second one or more documents to identify one or more questions stored in one



31

or more designated fields of one or more of: the identified first one or more documents, or the identified second one or more documents;

comparing the generated question with a first plurality of questions within a question-answer repository such that a first plurality of similarity scores are generated, each of the first plurality of similarity scores representing a similarity between the generated question and a corresponding one of the first plurality of questions;

determining whether the generated question is semantically distinct from the first plurality of questions based, at least in part, on the similarity scores;

adding the generated question to the question-answer repository based, at least in part, on having determined that the generated question is semantically distinct from the first plurality of questions; and

updating the set of documents to include the generated question.

**16.** The computer program product as recited in claim **15**, the program code further comprising computer-readable instructions configurable to cause:

    parsing the set of documents to extract one or more text segments;

    wherein generating the question is performed using the extracted text segments.

**17.** The computer program product as recited in claim **15**, the program code further comprising computer-readable instructions configurable to cause:

    training a question generation model using a set of question generation training data;

    wherein generating the question is performed using the question generation model.

**18.** The computer program product as recited in claim **15**, the program code further comprising computer-readable instructions configurable to cause:

32

obtaining a user question received from a client device;

comparing the user question with a second plurality of questions within the question-answer repository such that a second plurality of similarity scores are generated, each of the second plurality of similarity scores representing a similarity between the user question and a corresponding one of the second plurality of questions;

determining whether the generated question is semantically distinct from the second plurality of questions based, at least in part, on the second plurality of similarity scores; and

adding the user question to the question-answer repository based, at least in part, on a result of determining whether the user question is semantically distinct from the second plurality of questions.

**19.** The computer program product as recited in claim **18**, the program code further comprising computer-readable instructions configurable to cause:

    training a similarity function using a set of similarity training data;

    wherein comparing the generated question with the first plurality of questions within the question-answer repository and comparing the user question with the second plurality of questions within the question-answer repository are performed using the similarity function.

**20.** The computer program product as recited in claim **15**, the program code further comprising computer-readable instructions configurable to cause:

    obtaining an answer to the question; and

    updating the question-answer repository to include the answer such that the answer is associated with the question.

\* \* \* \* \*